Lezione 4
11 ottobre 1999

Argomenti trattati
- Strutture di controllo fondamentali: esempi.

4.1 Strutture di controllo fondamentali: esempi

Numeri pari e dispari

Vogliamo scrivere un algoritmo che, ricevendo un numero intero, indichi se esso è pari o dispari. La struttura dell'algoritmo si basa su una selezione:

```
leggi il numero n
calcola il resto della divisione di n per 2
SE il resto è uguale a zero
    ALLORA
        scrivi "pari"
    ALTRIMENTI
        scrivi "dispari"
FINESE
```

Riscriviamo ora l'algoritmo, introducendo due variabili `n` e `resto`, destinate a contenere numeri interi. Da qui in avanti, il calcolo del quoziente e del resto della divisione intera verranno indicando utilizzando gli operatori binari tra interi `DIV` e `MOD`. Ad esempio `5 DIV 3` vale `1`, mentre `5 MOD 3` vale `2`. Utilizziamo, inoltre, il simbolo `=` per indicare l'operatore di confronto "uguale".

```
variabili n, resto: numeri interi

leggi n
resto ← n MOD 2
SE resto = 0
    ALLORA
        scrivi "pari"
    ALTRIMENTI
        scrivi "dispari"
FINESE
```

©1999 Giovanni Pighizzini
Il contenuto di queste pagine è protetto dalle leggi sul copyright e dalle disposizioni dei trattati internazionali. Il titolo ed i copyright relativi alle pagine sono di proprietà dell'autore. Le pagine possono essere riprodotte ed utilizzate liberamente dagli studenti, dagli istituti di ricerca, scolastici ed universitari afferenti ai Ministeri della Pubblica Istruzione e dell'Università e della Ricerca Scientifica e Tecnologica per scopi istituzionali, non a fine di lucro. Ogni altro utilizzo o riproduzione (ivi incluse, ma non limitatamente a, le riproduzioni a mezzo stampa, su supporti magnetici o su reti di calcolatori) in toto o in parte è vietata, se non esplicitamente autorizzata per iscritto, a priori, da parte dell'autore.
L'informazione contenuta in queste pagine è ritenuta essere accurata alla data della pubblicazione. Essa è fornita per scopi meramente didattici e non per essere utilizzata in progetti di impianti, prodotti, ecc.
L'informazione contenuta in queste pagine è soggetta a cambiamenti senza preavviso. L'autore non si assume alcuna responsabilità per il contenuto di queste pagine (ivi incluse, ma non limitatamente a, la correttezza, completezza, applicabilità ed aggiornamento dell'informazione). In ogni caso non può essere dichiarata conformità all'informazione contenuta in queste pagine. In ogni caso questa nota di copyright non deve mai essere rimossa e deve essere riportata anche in utilizzi parziali.

Il calcolo del resto può essere effettuato direttamente al momento della valutazione della condizione della selezione, senza introdurre esplicitamente una variabile:

```
variabili n: numero intero

leggi n
SE n MOD 2 = 0
    ALLORA
        scrivi "pari"
    ALTRIMENTI
        scrivi "dispari"
FINESE
```

Calcolo della somma di una sequenza di numeri

Descriveremo ora degli algoritmi per trattare iterativamente sequenze di numeri. Come primo esempio consideriamo il problema del calcolo della somma di una sequenza di numeri interi. Un primo schema di soluzione è il seguente:

```
poni il valore della somma a zero
ESEGUI
    leggi un numero
    aggiungi il numero alla somma
FINO A QUANDO i numeri sono finiti
scrivi la somma
```

Come fa l'esecutore a sapere quando i numeri sono finiti? Un primo metodo, consiste nel comunicare inizialmente all'esecutore quanti sono i numeri che andranno letti. L'esecutore ripete le istruzioni del ciclo tante volte quanti sono i numeri:

```
leggi quanti numeri devi sommare
poni il valore della somma a zero
ESEGUI
    leggi un numero
    aggiungi il numero alla somma
FINO A QUANDO hai letto tutti i numeri
scrivi la somma
```

Osserviamo che l'algoritmo precedente richiede che ci sia sempre almeno un numero da sommare. Nel caso l'utente voglia sommare zero numeri, l'algoritmo non opera correttamente.

Questo problema può essere facilmente risolto, utilizzando un ciclo con il controllo in testa, cioè un ciclo del tipo `QUANDO ... RIPETI`. Ricordiamo che l'uscita da questo ciclo avviene quando la condizione risulta *falsa*. Pertanto, l'algoritmo può essere riscritto come:

```
leggi quanti numeri devi sommare
poni il valore della somma a zero
QUANDO c'è ancora qualche numero da leggere ESEGUI
    leggi un numero
    aggiungi il numero alla somma
RIPETI
scrivi la somma
```

Il controllo del ciclo precedente può essere realizzato introducendo una variabile per contare quanti numeri siano già stati letti. Tale variabile, posta a zero all'inizio dell'algoritmo, viene incrementata

ad ogni iterazione. Introduciamo inoltre una variabile in cui immagazzinare la somma, una variabile per memorizzare l'ultimo numero letto, e una variabile, di nome `quanti`, in cui memorizziamo il numero di numeri che devono essere letti:

```
variabili somma, numero, quanti, cont: numeri interi

leggi quanti
somma ← 0
cont ← 0
QUANDO cont < quanti ESEGUI
    leggi numero
    somma ← somma + numero
    cont ← cont + numero
RIPETI
scrivi somma
```

Nella soluzione precedente, l'utente è costretto a comunicare a priori al programma quanti sono i numeri da sommare. Questo può essere scomodo, soprattutto quando si abbia un lungo elenco. Sarebbe piú vantaggioso fissare un valore che indichi la fine della sequenza. Utilizziamo a questo scopo il numero zero. Pertanto, se verranno inseriti `5 8 7 0`, il risultato dovrà essere `20`, mentre se verrà inserito solamente `0` il risultato sarà `0`. Basandoci sul primo schema di algoritmo che abbiamo presentato per questo problema, possiamo scrivere:

```
poni il valore della somma a zero
ESEGUI
    leggi un numero
    aggiungi il numero alla somma
FINO A QUANDO il numero letto è zero
scrivi la somma
```

Osserviamo che il numero zero, che funge da indicatore di fine sequenza, viene aggiunto alla somma, prima di uscire dal ciclo. L'aggiunta di zero alla somma è inutile e, sebbene non modifichi il risultato, andrebbe evitata (ad esempio, se anziché calcolare la somma dovessimo calcolare il prodotto, la moltiplicazione per zero modificherebbe il risultato). Per fare ciò possiamo introdurre un costrutto di selezione, per eseguire l'operazione di somma solo quando il numero letto è diverso da zero:

```
poni il valore della somma a zero
ESEGUI
    leggi un numero
    SE il numero è diverso da zero
        ALLORA
            aggiungi il numero alla somma
    FINESE
FINO A QUANDO il numero letto è zero
scrivi la somma
```

Notiamo che l'istruzione leggi un numero viene eseguita sempre almeno una volta. Inoltre, ad ripetizione del ciclo, viene controllata due volte una condizione sul numero letto: la prima volta per la selezione, la seconda per l'iterazione. Per evitare questo doppio controllo, possiamo riscrivere l'algoritmo come segue:

```
poni il valore della somma a zero
leggi un numero
```

```
QUANDO il numero è diverso da zero ESEGUI
    aggiungi alla somma il numero
    leggi un numero
RIPETI
scrivi la somma
```

Dettagliamo l'algoritmo introducendo due variabili, una per memorizzare la somma, l'altra per memorizzare l'ultimo numero letto. Indichiamo con <> l'operatore di confronto "diverso":

```
variabili somma, numero: numeri interi

somma ← 0
leggi numero
QUANDO numero <> 0 ESEGUI
    somma ← somma + numero
    leggi numero
RIPETI
scrivi somma
```

Numeri pari e numeri dispari in una sequenza

Vogliamo ora descrivere un algoritmo che riceva dall'esterno una sequenza di numeri interi e indichi quanti di questi sono pari e quanti sono dispari. Supponiamo che l'inserimento di zero indichi la fine della sequenza (zero non dovrà, pertanto, essere considerato).

Osserviamo che l'ultimo algoritmo che abbiamo presentato è basato sulla seguente struttura:

```
...fase iniziale...
leggi un numero
QUANDO il numero è diverso da zero ESEGUI
    ...esamina il numero...
    leggi un numero
RIPETI
...fase finale...
```

Questa struttura può essere utilizzata ogni volta che si debba trattare una sequenza di numeri terminata da zero.

Nella ...fase iniziale... vengono assegnati dei valori alle quantità utilizzate nell'algoritmo. In questo caso verranno utilizzati due contatori, uno per i numeri pari, l'altro per i numeri dispari. Nella ...fase finale... i valori di questi contatori verranno comunicati all'esterno.

Analizziamo ora la fase ...esamina un numero..., in cui si deve controllare se il numero è pari o dispari, e incrementare il contatore opportuno:

```
SE il numero è pari
    ALLORA
        incrementa il contatore dei numeri pari
    ALTRIMENTI
        incrementa il contatore dei numeri dispari
FINESE
```

L'algoritmo completo è:

```
azzera i due contatori
leggi un numero
QUANDO il numero è diverso da zero ESEGUI
```

```
    SE il numero è pari
        ALLORA
            incrementa il contatore dei numeri pari
        ALTRIMENTI
            incrementa il contatore dei numeri dispari
    FINESE
    leggi un numero
RIPETI
scrivi i valori dei contatori
```

L'algoritmo può ora essere dettagliato, introducendo esplicitamente le variabili, le opportune istruzioni di assegnamento e le condizioni:

```
variabili numero, npari, ndispari: numeri interi
npari ← 0
ndispari ← 0
leggi numero
QUANDO numero <> 0 ESEGUI
    SE numero MOD 2 = 0
        ALLORA
            npari ← npari + 1
        ALTRIMENTI
            ndispari ← ndispari + 1
    FINESE
    leggi numero
RIPETI
scrivi npari, ndispari
```

Calcolo del massimo comun divisore

Nella prima lezione abbiamo presentato l'algoritmo di Euclide per il calcolo del massimo comun divisore tra due interi positivi. Abbiamo poi visto come riscrivere questo algoritmo in un linguaggio assembler. Riscriviamo ora lo stesso algoritmo utilizzando le strutture di controllo fondamentali. Per comodità riportiamo il testo dell'algoritmo:

1. Siano x e y i due numeri.
2. Calcola il resto della divisione di x per y.
3. Se il resto è diverso da zero, ricomincia dal passo 2
   utilizzando come x il valore attuale di y, e come y il valore del resto,
   altrimenti prosegui con il passo successivo.
4. Il massimo comun divisore è uguale al valore attuale di y.

Osserviamo che l'istruzione al passo 2 deve essere ripetuta un certo numero di volte, sulla base di una condizione che viene controllata al passo successivo. Dunque, il controllo avviene *dopo* avere eseguito l'istruzione da ripetere. Inoltre, prima dell'esecuzione del ciclo, occorre effettuare la lettura dei dati, mentre dopo l'uscita dal ciclo è necessario comunicare all'esterno il risultato. Pertanto la struttura dell'algoritmo è:

```
leggi i numeri x, y
ESEGUI
    ...blocco di istruzioni...
FINO A QUANDO il resto è uguale a zero
scrivi il valore di y
```

Definiamo ora il ...blocco di istruzioni.... All'interno di esso andrà effettuato il calcolo del resto della divisione. Successivamente, nel caso il resto sia diverso da zero, andranno predisposti i nuovi valori di x e di y da utilizzare nell'iterazione successiva, secondo quanto specificato al passo 2. Introduciamo pertanto un costrutto di selezione, che effettua lo spostamento solo se il resto è diverso da zero:

```
calcola il resto della divisione di x per y
SE il resto è diverso da zero
    ALLORA
        utilizza come x il valore attuale di y e
        come y il valore del resto
FINESE
```

L'algoritmo completo è dunque:

```
leggi i numeri x, y
ESEGUI
    calcola il resto della divisione di x per y
    SE il resto è diverso da zero
        ALLORA
            utilizza come x il valore attuale di y e
            come y il valore del resto
    FINESE
FINO A QUANDO il resto è uguale a zero
scrivi il valore di y
```

Riscriviamo ora l'algoritmo introducendo tre variabili, che chiameremo `x`, `y`, `resto`, destinate a contenere numeri interi.

```
variabili x, y, resto: numeri interi

leggi x, y
ESEGUI
    resto ← x MOD y
    SE resto <> 0
        ALLORA
            x ← y
            y ← resto
    FINESE
FINO A QUANDO resto = 0
scrivi y
```

Notiamo che lo spostamento dei valori, dopo il calcolo del resto, può essere effettuato anche quando il resto vale zero. In questo modo si evita di effettuare, ad ogni ciclo, un doppio controllo sul valore della variabile `resto`. Tuttavia, a causa dello spostamento, il risultato finale si troverà nella variabile `x`:

```
variabili x, y, resto: numeri interi

leggi x, y
ESEGUI
    resto ← x MOD y
    x ← y
    y ← resto
FINO A QUANDO resto = 0
scrivi x
```

Esercizi

1. Scrivere un algoritmo, basato sulle tre strutture di controllo fondamentali, che riceva in ingresso una sequenza di numeri e produca in uscita il prodotto dei numeri letti. Si supponga che l'inserimento del numero zero indichi la fine della sequenza.

2. Scrivere un algoritmo, basato sulle tre strutture di controllo fondamentali, che riceva in ingresso una sequenza di numeri interi e produca in uscita la somma dei numeri pari e la somma dei numeri dispari contenuti della sequenza. Si supponga che l'inserimento del numero zero indichi la fine della sequenza.

3. Scrivere un algoritmo, basato sulle tre strutture di controllo fondamentali, che riceva in ingresso una sequenza di numeri interi e produca in uscita la somma dei numeri di posto pari e la somma dei numeri di posto dispari contenuti della sequenza. Si supponga che l'inserimento del numero zero indichi la fine della sequenza.

4. Scrivere un algoritmo, basato sulle tre strutture di controllo fondamentali, che riceva in ingresso una sequenza di numeri e produca in uscita la media dei numeri letti. Si supponga che l'inserimento del numero zero indichi la fine della sequenza.

5. Scrivere un algoritmo, basato sulle tre strutture di controllo fondamentali, che riceva in ingresso una sequenza di numeri interi e riporti in uscita il piú grande e piú piccolo tra i numeri letti. Si supponga che l'inserimento del numero zero indichi la fine della sequenza.

6. Scrivere un algoritmo, basato sulle tre strutture di controllo fondamentali, per il calcolo del prodotto di due fattori interi positivi mediante la seguente tecnica: a ogni passo si raddoppia il primo fattore e si dimezza il secondo, terminando quando il secondo fattore diventa `1`. Durante questo procedimento si sommano tra loro i primi fattori corrispondenti a secondi fattori dispari. Il valore finale di tale somma è uguale al prodotto da calcolare. Ad esempio, per il calcolo del prodotto tra `34` e `21`, si effettueranno i seguenti passi:

primo fattore	secondo fattore	numero da sommare
34	21	34
68	10	
136	5	136
272	2	
544	1	544

   La somma dei valori della terza colonna è uguale al prodotto dei due numeri dati.