Procedimenti e funzioni

Siamo arrivati al terzo tipo di struttura di programmazione: le subroutines, che si dividono nella maggioranza dei linguaggi in procedimenti e funzioni.

Utilità delle subroutines

Anche se il modo di definire le subroutines è diverso, a seconda del linguaggio, l'utilità è comune per tutti. Possiamo vedere ciò da due punti di vista. Normalmente un problema complesso si può scomporre in diversi sotto-problemi di minore difficoltà, questo oltre a semplificare un programma lungo, ne migliora l'organizzazione. Questi sotto-problemi sono risolti dalle subroutines o subalgoritmi. Una subroutine è una specie di scatola nera che riceve i dati, realizza le operazioni e ottiene dei risultati. Questo facilita la programmazione dei grandi progetti, e la programmazione in gruppi di differenti subroutines che più tardi saranno messe insieme conoscendo unicamente le loro funzioni e i loro parametri di ingresso e uscita. Una subroutine, inoltre, si scrive una sola volta e dopo è possibile far riferimento a essa ("chiamarla") dagli altri punti del programma, e anche dall'interno di altre subroutines. Questo è un particolare molto interessante, perché di solito molti lavori specifici vengono eseguiti diverse volte in un programma, per cui si genererebbe un eccesso di codice. Con l'uso di subroutine si ottiene la riutilizzazione del codice, non solo nello stesso programma, ma anche in progetti indipendenti, risparmiando spazio nella memoria di programma e tempo di sviluppo. L'indipendenza fra i moduli permette inoltre una verifica e una correzione degli errori in modo più semplice.

`PROGRAM Cilindro;`
`CONST`
`  pi = 3.1416;`
`VAR`
`  raggio, altezza, superficie, volume: REAL;`
`BEGIN`
	Inizio del programma e dichiarazione di variabili

`  WRITELN('Introdurre il raggio e l'altezza: ');` `  READLN(raggio, altezza);`	Inserimento dei dati
`  superficie:=2*pi*raggio*altezza + 2*pi*SQR(raggio);` `  volume:=pi*SQR(raggio)*altezza;`	Operazioni
`  WRITELN('La superficie è: ',superf:8:2);` `  WRITELN('Il volume è: ',volume:8:2)`	Visualizzazione dei dati
`END.`	Fine del programma

Un problema è formato da diversi sottoproblemi.

Formazione di subroutines

Qualche dubbio però sorge sempre: come si realizza questa suddivisione? Sino a quando suddividere? Come dividere le subroutines in modo che siano riutilizzabili? La soluzione non è unica, e solo il tempo e la pratica forniranno risposte ottimali. Per iniziare si può applicare ciò che già si conosce riguardo agli organigrammi. Si possono realizzare differenti livelli di raggruppamenti, in modo che un riquadro possa essere costituito da una sola istruzione, un gruppo di queste o avere al suo interno strutture ripetitive e/o condizionali; in definitiva, ogni riquadro potrebbe essere una subroutine. Si tratta di trovare un punto di equilibrio, e se un determinato riquadro risulta troppo complesso o troppo semplice, si può vedere se è possibile la sua separazione o il suo raggruppamento all'interno di altri riquadri. Per quanto riguarda la riutilizzazione, ci sono parti di

In microrobotica c'è un alto grado di riutilizzo delle subroutines.

```
Programma                                   ' Inizio programma

leggere numero                              ' Si legge un numero dalla tastiera
mentre numero < = 0                         ' Si verifica che sia positivo
        scrivere "Introdurre un numero positivo"
        leggere numero
fine-mentre

risultato = RADICEQUADRATA(numero)          ' Si prende il suo valore e si chiama
                                            ' la funzione
scrivere"La radice quadrata di: ", numero, " è: ",   ' Si utilizza il valore acquisito
risultato

fine-programma                              ' Fine del programma principale

Funzione RADICEQUADRATA (valore)            ' Intestazione della funzione

radice = valore ^ (1/2)                     ' Calcolo del risultato
RADICEQUADRATA = radice                     ' Restituzione del risultato

fine-funzione                               ' Fine della funzione
```

Calcolo della radice quadrata utilizzando una funzione.

programma che sono più riutilizzabili di altre, o che si possono incontrare più o meno facilmente lungo lo sviluppo del programma stesso. In microrobotica sono molto comuni le routines di "temporizzazione", "rotazione per gradi", "avanzamento di z centimetri", ecc. Il passo successivo consiste nel pensare alle cose che vogliamo che faccia la subroutine, ai dati che vogliamo fornirle e a quelli che vogliamo ci restituisca, per fare in modo che

```
Programma                                   ' Inizio del programma

leggere numero                              ' Si legge un numero dalla tastiera
mentre numero < = 0                         ' Si verifica che sia positivo
        scrivere "Introdurre un numero positivo"
        leggere numero
fine-mentre

RADICEQUADRATA (numero)                     ' Si chiama il procedimento

fine-programma                              ' Fine del programma principale

Procedimento RADICEQUADRATA (valore)        ' Intestazione del procedimento

radice = valore ^ (1/2)                     ' Calcolo del risultato

scrivere "La radice quadrata di: ", valore, " è: ", radice   ' Si utilizza il risultato

fine-procedimento                           ' Fine del procedimento
```

Calcolo della radice quadrata utilizzando un procedimento.

abbia l'indipendenza e la funzionalità che la devono caratterizzare. Abbiamo detto che dall'interno di una subroutine se ne può chiamare un'altra; in questo caso dobbiamo seguire le regole del linguaggio di programmazione con cui stiamo lavorando, dato che alcuni di questi linguaggi prevedono che una subroutine debba prima essere definita e poi chiamata, inoltre bisogna fare attenzione all'ordine di chiamata delle varie subroutines. In altri linguaggi questo non è importante e le subroutines, di solito, vengono messe per convenzione alla fine del programma. Ci sono anche linguaggi dove le subroutines sono programmate su videate indipendenti.

Differenze tra procedimenti e funzioni

Le subroutines possono essere di due tipi: procedimenti e funzioni. Quanto detto finora è valido per entrambi. Infatti entrambi possono accettare dati, richiamare argomenti o parametri per eseguire il loro lavoro. Tuttavia a differenza del procedimento, la funzione restituirà un valore. Questo fa sì che si utilizzi una funzione oppure un procedimento a seconda del caso. Immaginiamo di dover realizzare la radice quadrata di un numero. La cosa più sensata è comporre l'algoritmo come una funzione, che ci restituirà la radice quadrata per essere visualizzata, sommata a un altro numero, memorizzata, ecc. Se invece utilizziamo un procedimento al posto di una funzione, sarà all'interno del procedimento che avverrà la visualizzazione, la somma, ecc. e lo dovremo modificare ogni volta che avremo bisogno di qualcosa di diverso. Inoltre, il risultato di una funzione, deve essere assegnato a una variabile, o inserito in una espressione, altrimenti il valore andrebbe perso. Comunque, se volessimo fare una routine di "temporizzazione", non ci interesserebbe un eventuale risultato come risposta, ma solo che il sistema "perda tempo", quindi in questo caso, sarebbe più corretto utilizzare un procedimento.