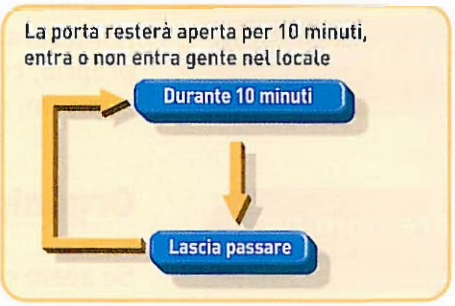
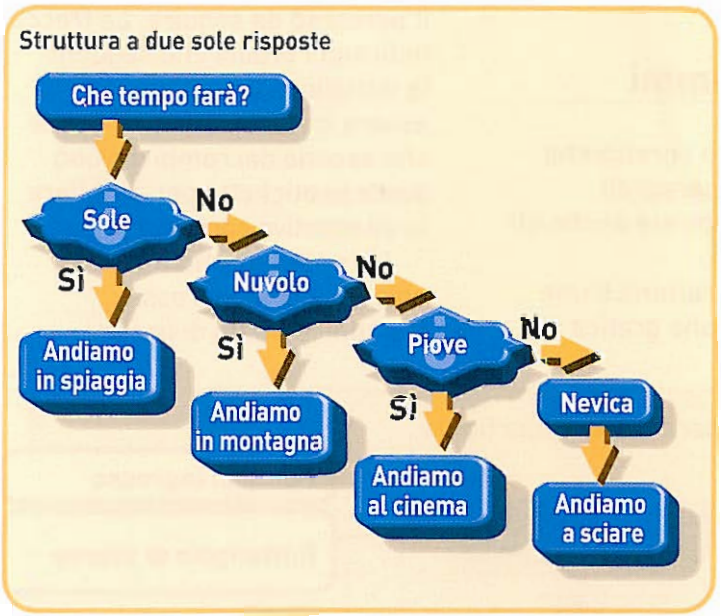




Due differenti strutture di controllo ripetitive per lo stesso processo.

Bisogna saper eseguire le conversioni fra le differenti strutture di controllo condizionali.

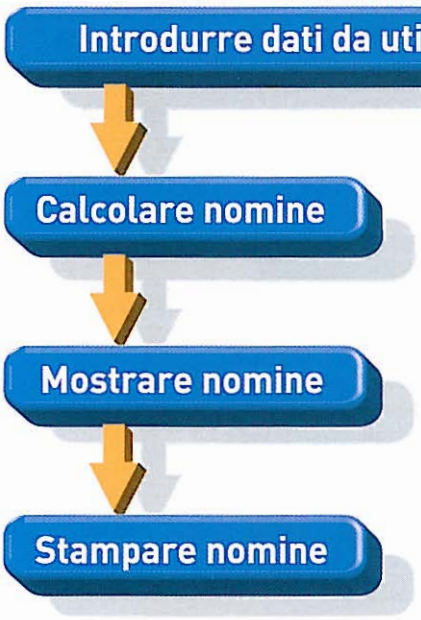
numero di volte noto oppure sconosciuto, o le strutture ripetitive. In definitiva, un programma può sembrare un albero con diversi rami, in cui non si conosce a priori quante istruzioni si dovranno eseguire né per quante volte.

Il caso più complesso di una struttura condizionale è quello con risposte multiple ad una singola domanda. Per ogni risposta esiste una sequenza di azioni o istruzioni da eseguire; si tratta della condizione chiamata alternativa a rami multipli. Se il numero di risposte si riduce a due (Sì o No, Vero o Falso) siamo nel caso di una condizionale semplice. Normalmente un linguaggio di programmazione possiede entrambe le strutture di controllo, in caso contrario dovremo saper fare la conversione da una all'altra, ma si tratta di una cosa relativamente semplice da realizzare.

La ripetizione di un insieme di istruzioni può anche avvenire sotto determinate condizioni; ad esempio un numero fisso di volte, oppure mentre l'utente non preme nessun tasto, sino a che si arriva ad un valore prefissato, sino a che si compia una determinata condizione, o durante quest'ultima, ecc. Ogni linguaggio avrà anche differenti forme per esprimere queste strutture, che sono sempre intercambiabili fra loro con piccoli aggiustamenti.

Subroutines

Abbiamo visto che le istruzioni si organizzano per essere eseguite mediante le strutture di controllo. Non è raro inoltre trovare, all'interno di un programma, una struttura dentro l'altra, in modo che all'interno di una condizionale ci sia una ripetitiva



Programma principale formato da quattro subroutines, ognuna delle quali avrà un processo interno.

o viceversa, oltre alle strutture sequenziali; oppure trovare addirittura sequenze di strutture. Che cosa accade invece, quando il programma è molto grande o ci sono parti di esso che devono essere ripetute in diversi punti? Per questi casi si utilizzano le subroutines, già note in alcuni linguaggi con il nome di procedimenti o funzioni, che non sono altro che frammenti di programma, raggruppati sotto lo stesso nome.

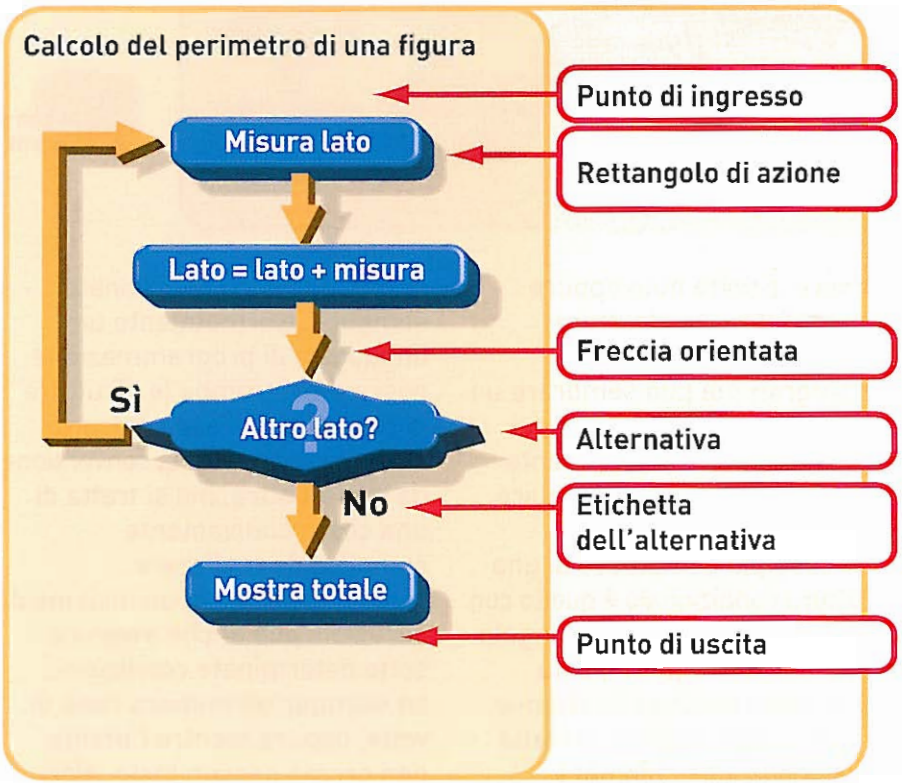
Quando si vorrà eseguire quel pezzo di programma, non sarà necessario scriverlo per intero nuovamente, ma basterà far riferimento al suo nome, fare cioè una chiamata alla subroutine. A seconda del linguaggio sarà necessario passare dei parametri alla subroutine per farle realizzare il suo compito. L'utilizzo di subroutines, oltre a far risparmiare il codice, dato che non è necessario riscriverlo, rende i programmi "più puliti": ci sarà un programma principale che avrà il compito di chiamare le subroutines e queste realizzeranno il lavoro vero e proprio, con cicli, condizioni, algoritmi di calcolo, ecc.

Organigrammi

Se avete capito i grafici che illustravano i paragrafi precedenti, capirete anche gli organigrammi.

Un organigramma è una rappresentazione grafica di quello che fa un programma. Dobbiamo considerare alcune norme di base che bisogna seguire, perché sia un linguaggio universale, indipendentemente da come si procederà dopo. È il primo passo per fare un programma, il meno complesso. Gli organigrammi sono formati da una serie di rettangoli, che contengono le azioni da eseguire e da rombi che contengono le condizioni che determinano il percorso da seguire. Le frecce indicano l'ordine che seguono le istruzioni, quindi devono essere orientate, inoltre quelle che escono dai rombi devono avere le etichette per scegliere le alternative correttamente.

Un organigramma ha un unico punto di ingresso e un unico punto di uscita.



Organigramma tipico che soddisfa le norme generali.