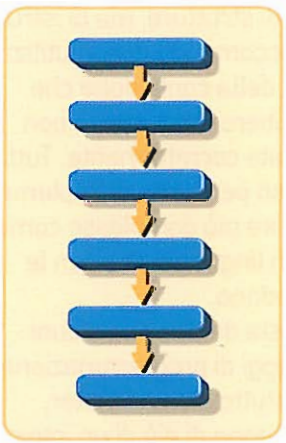
# Salti condizionati e incondizionati

In un capitolo precedente abbiamo classificato le strutture di controllo come strutture condizionali e cicli. Tuttavia, secondo il linguaggio di programmazione con cui si sta lavorando, è possibile trovare altri tipi di classificazione che paiono essere in contrasto fra i loro su alcuni concetti che erano già stati dati per scontati.

Ad esempio, alcuni libri o manuali segnalano che le strutture si classificano in condizionali, incondizionali, di chiamata e di ritorno da subroutine, e che tutte sono necessarie.

Altri raccomandano di non utilizzare strutture incondizionali o istruzioni di salto incondizionato.

Ma allora, chi ha ragione? Vediamo di distinguere tutti questi termini, per classificarli secondo il loro uso, anche se il modo di denominarli non è sempre il più adeguato.

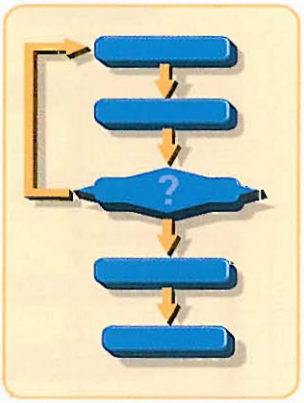


Rappresentazione di un programma che si esegue in sequenza.

## Strutture condizionali e incondizionali

Partiamo dal concetto che abbiamo già avuto modo di commentare, cioè che raramente un programma sarà una sequenza di istruzioni eseguite una dietro l'altra. Nel caso fosse così lo potremmo rappresentare come mostrato nella figura, dove ogni riquadro è formato da un insieme di istruzioni che hanno una qualche relazione fra loro.

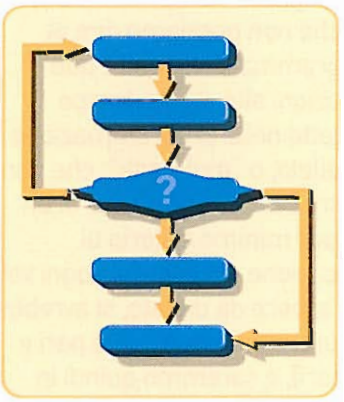
Tuttavia, un programma con strutture condizionali, avrebbe la forma della seconda figura, dove i rombi rappresentano le condizioni che esigono una scelta fra due



Rappresentazione di un programma con strutture condizionali.

opzioni. Alcuni linguaggi hanno istruzioni che permettono la scelta solo fra due percorsi, altri invece hanno istruzioni che permettono scelte multiple, come la struttura rappresentata nella terza figura.

Come si può vedere, ogni volta che si salta da una parte all'altra



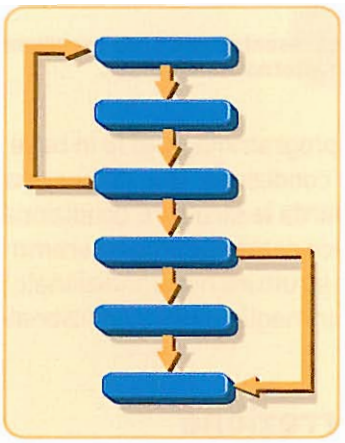
Rappresentazione di un programma con alternative multiple.

del programma, lo si fa in base a una condizione. Questo per quanto riguarda le strutture condizionali; invece come rappresenteremmo una struttura non condizionale? O per meglio dire incondizionale?

## Correzione di un programma

Conosciamo già questo tipo di nomenclatura, dato che la utilizziamo per fare gli organigrammi dei nostri programmi. Come certamente ricorderete esistono delle norme da seguire al momento di realizzare gli organigrammi, ad esempio da un rettangolo può uscire una sola freccia per indicare la sequenza, invece da un rombo, posto che sia utilizzato in alternativa, usciranno almeno due frecce. Pertanto se l'organigramma mostra l'aspetto della figura della pagina successiva, significa che è stato mal costruito. Quando si arriva al

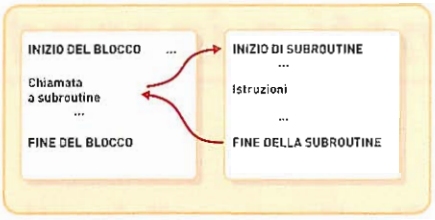
rettangolo con due frecce che percorso bisogna seguire? Se non c'è una condizione che ci permette di decidere, si tratta di incondizionale, quindi questo organigramma non ha senso, perché non possiamo dire al programma di eseguire due istruzioni allo stesso tempo (eccetto nella programmazione in parallelo, o "multitask", che non è il nostro caso). In presenza di un seppur minimo criterio di esecuzione, ad esempio ogni volta che si esce da un lato, si avrebbe già una condizione (volte pari e dispari), e saremmo quindi in



**Rappresentazione di un programma con struttura incondizionale.**

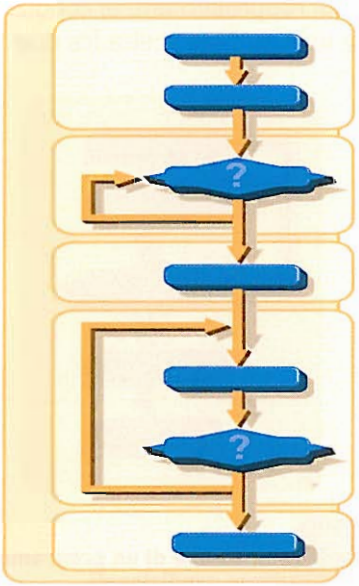
presenza di una struttura condizionale. Bisogna perciò aver chiaro che le strutture non condizionali, nei nostri programmi, non esistono. Quando si realizza un salto bisogna che questo sia basato su di una condizione, tuttavia non bisogna confondere un salto con una chiamata a subroutine.

Abbiamo già visto che quando si chiama una subroutine, praticamente si esegue un salto a un'altra parte del programma per eseguire un insieme di istruzioni, dopo di che si ritorna al punto di




**Procedura che si segue chiamando una subroutine.**

partenza per continuare la sequenza; quindi all'interno di un organigramma la chiamata a subroutine viene rappresentata con un riquadro, come un'istruzione normale. I cicli devono essere considerati come strutture cicliche o iterative, separati dalle strutture condizionali o incondizionali, però se devono essere inclusi in una di queste due categorie saranno anche loro condizionali, dato che



**Un programma corretto deve poter essere conglobato in un unico riquadro.**

dipendono da una variabile, anche se di tipo contatore, come nella FOR, per uscire dalla iterazione.

In ogni caso deve essere sempre possibile poter conglobare le diverse strutture di un organigramma in strutture più complesse, fino ad arrivare a un unico riquadro. I riquadri non si possono incrociare fra loro, e le uniche frecce che possono uscire da un riquadro verso un altro sono quelle di sequenza.

## Istruzione di salto condizionato e incondizionato

Anche se in alcuni manuali vengono denominati nello stesso modo, una cosa sono le strutture incondizionali, un'altra le istruzioni di salto incondizionato. Abbiamo già detto che le prime, per principio, non sono possibili, però le istruzioni incondizionali non solo sono utili, ma in alcuni linguaggi sono indispensabili. È il caso ad esempio dell'istruzione GOTO, che anche se incondizionale, si utilizza in strutture condizionali, accompagnando, ad esempio, la IF, per deviare da una parte del programma verso un'altra.

Questo è ciò che si intende quando si dice che non si utilizzano strutture non condizionali; non si tratta di strutture, ma di istruzioni, e si raccomanda di non utilizzarle a causa della confusione che potrebbero produrre se non utilizzate correttamente. Tuttavia, il loro uso permette di implementare strutture più complesse come i cicli, in linguaggi che non le possiedono.

Resta da dire che alcuni linguaggi di programmazione, soprattutto gli assembler, dispongono di più di un'istruzione di salto incondizionato, per poter fare, ad esempio, "salti brevi", "salti vicini" e "salti lontani".