un'altra, ma un gruppo di istruzioni oppure un altro. Inoltre permette di non realizzare tutte le comparazioni, nelle strutture che abbiamo visto in precedenza in cui si utilizza la struttura IF...THEN come base.

```
IF A>B THEN GOTO Label
           Inst1
           Inst2
           Inst3
GOTO Loop
Label:     Inst4
           Inst5
           Inst6
Loop:      Inst7
           Inst8
```

Struttura IF...THEN insieme alla GOTO.

Nell'esempio della figura, se si compie la condizione, si va all'etichetta "Label", con la quale si eseguono le istruzioni dalla 4 alla 6. Se la condizione non si compie si prosegue in sequenza, eseguendo le istruzioni dalla 1 alla 3 e in seguito si salta a "Loop", in modo che la parte di "Label" non venga eseguita. Le istruzioni 7 e 8 si eseguono sempre.

Le subroutines

Una subroutine, così come un salto incondizionato con l'istruzione GOTO, suppone una rottura della sequenza del programma, perché si salta ad un'altra parte del codice per eseguire un gruppo di istruzioni. La differenza con la struttura GOTO è che quest'ultima non ritorna alla posizione originale da cui è partita, invece chiamando una subroutine con GOSUB si salta ad un indirizzo segnato da "etichetta" e, dopo aver eseguito una serie di istruzioni, tramite l'istruzione RETURN, si torna all'istruzione successiva da cui si è chiamata la subroutine per proseguire con il programma principale.

```
GOSUB    {etichetta}
         ...
         ...
RETURN
```

Sintassidell'esecuzione di subroutine.

Nell'esempio, prima si fa una chiamata alla subroutine "Esci", si eseguono le istruzioni 1 e 2 e si ritorna dopo aver trovato l'istruzione RETURN. In seguito si chiama la subroutine "Subb", con cui si eseguono le istruzioni 3, 4 e 5 e nuovamente si ritorna con l'istruzione RETURN. Se si ha l'accortezza di posizionare le routines alla fine del programma e si pone un'istruzione STOP davanti alla prima, il programma si fermerà qui ed entrerà in un ciclo infinito, in modo che non arriverà alle subroutines senza una chiamata specifica con GOSUB.

```
         GOSUB Esci
         GOSUB Subb
Esci:    Inst1
         Inst2
         RETURN
Subb:    Inst3
         Inst4
         Inst5
RETURN
```

Esempio dell'esecuzione di subroutine.

I cicli FOR

Abbiamo già visto diversi modi di rompere la sequenza del programma, eseguendo solo le istruzioni che interessano nei vari casi. Queste istruzioni di solito si raggruppano in modo che l'insieme si possa ripetere un numero definito di volte, senza doverle copiare nuovamente nel codice.
Nel LetPicBasic per questa funzione si utilizzano i cicli FOR, la cui sintassi è mostrata nella figura.

"Variabile" si incrementerà dal "valore_iniziale" sino al "valore_finale". Se si utilizza il comando STEP, che è opzionale, "passo_salto" indica il valore con cui si incrementa la variabile prima di arrivare al codice. "Codice" saranno le istruzioni da eseguire, e la variabile si incrementa quando si arriva a "STEP"; se non si è raggiunto il limite si tornerà ad eseguire il

```
A=3
FOR X=1 TO 10 STEP 2
A=A+X
NEXT X
```

Esempio con la struttura FOR...NEXT.

```
FOR {variabile} = {valore_iniziale} TO {valore_finale} [STEP] {passo_salto}
        {codice}
NEXT {variabile}
```

Sintassi della struttura FOR...NEXT.

codice, e se si è raggiunto si uscirà dal ciclo per continuare il programma.
Nell'esempio, il ciclo si incrementa di 2 in 2 passi, per cui si esegue 5 volte. Alla variabile A, che parte con valore 3, verrà sommato il valore di X ogni volta. Con ogni istruzione NEXT, se X non è arrivato al valore finale, si ritorna all'inizio del ciclo per ripeterlo utilizzando però il valore aggiornato di X.
È possibile annidare diversi cicli uno dentro l'altro, tenendo conto che il totale del numero di volte che si eseguono le istruzioni centrali sarà la moltiplicazione del numero di volte che si esegue un ciclo, per il numero di volte che si esegue un altro ciclo, e così via a seconda degli annidamenti che si realizzano.

```
A=0
B=4
FOR X=0 TO 4 STEP 2
FOR Y=B TO 20
    A=X+Y
    B=X-Y
NEXT Y
NEXT X
```

Esempio di strutture FOR...NEXT annidate.

Ricordate che è necessario che i cicli si trovino uno all'interno dell'altro; quello centrale, che utilizza la variabile "Y", inizia con un "FOR Y..." e termina con un "NEXT Y", e il ciclo che utilizza la variabile "X" le ingloba completamente, iniziando prima e terminando dopo. Le istruzioni NEXT non potranno essere scambiate l'una per l'altra.

Altre strutture di controllo

Così come succedeva con alcune modalità di "IF", il LetPicBasic non dispone di tutte le strutture esistenti negli altri linguaggi di programmazione, per fare dei cicli. Dovremo quindi adattare quelle che già conosciamo per realizzare le stesse funzioni. Così ad esempio, nel caso si voglia ripetere una sequenza di istruzioni "mentre" si compie una certa condizione, bisogna utilizzare la sentenza "WHILE". Quando la condizione cessa di compiersi il programma continua con la successiva istruzione alla "WHILE". L'istruzione "REPEAT" è simile alla precedente, ma la valutazione della condizione si realizza alla fine, quindi le istruzioni si eseguono almeno una volta, e si ripetono sino a che la condizione si compie. In entrambi i casi si utilizzano istruzioni IF...THEN e GOTO per formare queste strutture.

```
Etichetta 1: IF {no-condizione} THEN GOTO Etichetta 2
             Inst1
             Inst2
             ...
             GOTO Etichetta 1
Etichetta 2:
             ...
```

Sintassi della struttura WHILE a partire da strutture IF...THEN e GOTO.

```
Etichetta1:
             Inst1
             Inst2
             ...
IF {no-condizione} THEN GOTO Etichetta1
Etichetta2:
             ...
```

Sintassi della struttura REPEAT a partire da strutture IF...THEN e GOTO.