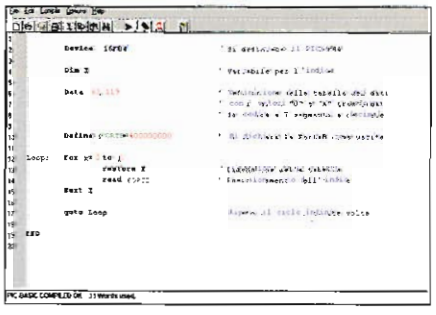
Regole di programmazione

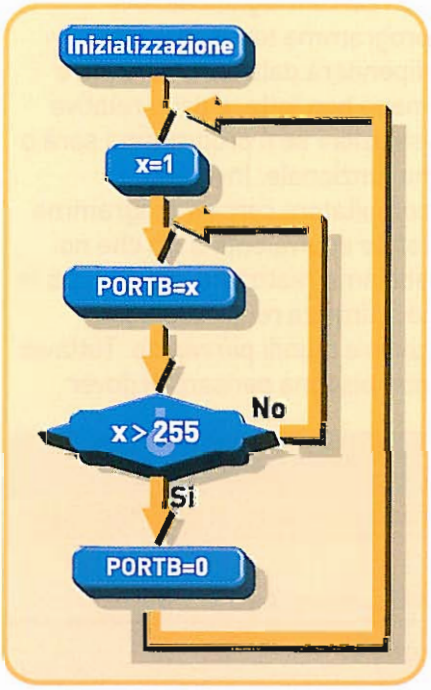
Abbiamo già realizzato diversi esercizi di prova per le istruzioni, però non abbiamo ancora affrontato programmi importanti, inoltre abbiamo già visto e seguito alcuni concetti al momento di programmare e ora vogliamo rimarcare l'importanza che riveste il seguire delle regole: stiamo parlando di Metodologia della Programmazione; un buon programmatore segue sempre dei modelli che aiutano a strutturare il programma in modo tale che sia facilmente comprensibile e modificabile dagli altri. Non bisogna far altro che guardare la figura in alto per capire il perché della metodologia. Difficilmente riusciremo ad associare quel piccolo programma a qualcosa che abbiamo già visto da altre parti, e poi la sua comprensione non è affatto immediata, nonostante sia composto da poche istruzioni; immaginate ciò che capiterebbe con un programma più lungo.



Programma che segue una metodologia di programmazione.

Programma che non segue la metodologia di programmazione.

La causa di ciò risiede in una regola non seguita — raccomandabile anche se non obbligatoria — cioè quella che abbiamo appena menzionato: strutturare il codice sia come tabulazioni, commenti, etichette, ecc.



Fasi nella risoluzione di un programma.

Il linguaggio di programmazione

Abbiamo già visto che pur con piccole modifiche, esistono sempre concetti e parti comuni in tutti i programmi. Il linguaggio di programmazione è l'insieme di regole, note, simboli e/o caratteri che permettono al programmatore di esprimere la risoluzione di un problema sotto forma di programma comprensibile dal sistema che lo dovrà eseguire. Una buona tecnica è plasmarlo prima su un organigramma, poi trasformarlo in codice indipendente dal linguaggio, quello che si chiama pseudo-codice, infine terminare utilizzando la sintassi e la grammatica proprie del linguaggio corrispondente. Tuttavia, non bisogna dimenticare che noi lavoriamo con linguaggi

```
INIZIO
      X=1
      Ripetere 255 volte
          PORTB=x
      Fine-Ripetere
      PORTB=0
      tornare a INIZIO
```

applicati a sistemi, che sono diversi dai linguaggi per il calcolo numerico (Fortran, Matlab), gestione (Cobol) o intelligenza artificiale (Prolog, Lisp), per citare solo alcuni esempi, che sono chiamati "linguaggi imperativi classici", e non funzionali basati su regole, orientati ad oggetti, ecc. dove si utilizza un altro tipo di programmazione e una metodologia propria per ognuno di essi. Più avanti passeremo da un linguaggio di alto livello a uno di basso livello come l'assembler.

Errori di lessico, sintassi e semantici

Il modo di risolvere gli errori commessi dipende dalla buona metodologia e dal tipo di errore: lessicale, di sintassi o semantico.

Quando compiliamo un programma in LetPicBasic, esso viene tradotto in codice macchina. Per prima cosa viene fatta un'analisi, cioè una verifica che il programma scritto, il programma sorgente, soddisfi le regole di definizione del linguaggio a tutti i livelli, tuttavia ci sono cose che non si possono provare automaticamente; il linguaggio non sa ciò che noi vogliamo fare, quindi non sa se le istruzioni utilizzate sono quelle adeguate o se avremmo dovuto utilizzarne altre. Per questo è importante avere una buona base metodologica, la quale verrà acquisita a poco a poco grazie agli esempi che vi proporremo. Se non ci sono errori (se ci sono non si va avanti) si realizza la traduzione. Tutto questo è fatto da un compilatore o da un interprete. Un interprete traduce il codice sorgente, e trasforma ogni istruzione di alto livello in diverse istruzioni di basso livello e le esegue. Sono metodi lenti, però permettono la modifica del programma durante l'esecuzione. I compilatori realizzano tutta la traduzione, dopo di che eseguono il programma, quindi la volta successiva che si esegue il programma non sarà necessaria la traduzione: sono più rapidi.

Per come lavora, un interprete non cerca il significato del programma totale, per cui dipenderà dalle strutture più o meno ben fatte, e dalle relative istruzioni se il programma sarà o no funzionale. Invece il compilatore cerca il programma totale equivalente a ciò che noi abbiamo realizzato, e se possibile lo ottimizza rendendolo più corto e quindi più rapido. Tuttavia non bisogna pensare di dover risparmiare linee di codice a tutti i costi; una buona struttura con più linee è migliore di un programma incomprensibile.
Esistono traduttori ibridi che riuniscono i vantaggi degli interpreti e dei compilatori.

LIVELLI	COSA SONO
Lessico	I simboli che si possono utilizzare in un linguaggio.
Sintassi	Le strutture valide che si possono formare con questi simboli.
Semantico	Significato della strutture.

Livelli di un linguaggio di programmazione.

Metodologia nelle aziende

Se per noi è importante seguire una metodologia, immaginatevi un prodotto grande sviluppato in diversi anni da più di un'analista e da più di un programmatore. Diventa imprescindibile stabilire delle regole comuni, in quanto a stili di codice e documentazione generata, in modo che tutti "parlino" la stessa lingua; il risultato sarà la riduzione di errori e la facilità di manutenzione.

Esistono numerose proposte di stili pubblicate per differenti linguaggi, anche se le stesse aziende di solito hanno un proprio capitolato. Identificare con i commenti l'autore e l'ultima data di modifica, l'uso di tutte le variabili, tabulare adeguatamente e commentare le routine, i salti, l'uso delle maiuscole e minuscole, ecc., sono alcuni esempi. Alcune operazioni come la tabulazione, la differenziazione dei tipi con i colori e altre simili, possono essere automatizzate con un buon editor. Per verificare la qualità dei programmi si utilizzano metodi di misura basati su alcuni parametri, quali la relazione fra le linee di codice sorgente e le linee di commento, linee per modulo o profondità di annidamento di una funzione, controllando tutto questo con la metodologia.