Elementi di linguaggi di programmazione
Manipolazione di simboli

Un calcolatore manipola sequenze di caratteri senza comprenderle.

Le regole secondo le quali avviene la manipolazione costituiscono il programma che il calcolatore esegue.

Le regole - i programmi - sono scritte in un linguaggio formale artificiale (cioè progettato espressamente per questo scopo), nel quale le modalità di manipolazione sono esprimibili in modo semplice, chiaro e non ambiguo.

Linguaggio di programmazione:

un linguaggio formale nel quale esprimere comandi e **flusso di controllo**

per esempio:

```
Leggi x; Per 10 volte trasforma x in x*x; Stampa x;
```

L'esecutore di uno specifico linguaggio di programmazione è la macchina astratta di quel linguaggio.

Gerarchie di linguaggi di programmazione

Lo *hardware* ha un suo linguaggio di programmazione per il quale la macchina astratta coincide con quella concreta. è il **linguaggio macchina** (di quello specifico *hardware*).

Il linguaggio macchina manipola direttamente le sequenze di bit fornite dall'*hardware*, utilizzando le operazioni primitive dell'*hardware* stesso (operazioni aritmetiche, salti).

Programmare in linguaggio macchina è faticoso, costoso ed espone ad errori.

Fin dagli anni 50 sono stati sviluppati linguaggi più evoluti, ciascuno dei quali progettato sopra un linguaggio più rudimentale.

Possiamo suddividere i linguaggi di programmazione in livelli, a seconda di quanto essi sono vicini al linguaggio macchina.

Basso livello:

1. **Linguaggio macchina** Le operazioni disponibili sono quelle direttamente fornite dall'*hardware*; ogni operazione è codificata da una sequenza di bit; ogni dato è indicato dall'indirizzo binario della parola di memoria in cui è memorizzato. Ogni indirizzo è espresso in modo assoluto (rispetto a tutta la memoria disponibile). Un programma è una sequenza di bit, che viene direttamente interpretata dall'*hardware*.
2. **Linguaggio assembler** (o assemblativo) Le operazioni sono quelle direttamente fornite dall'*hardware*, ma sono indicate da nomi convenzionali (mnemonici); i dati sono indicati da nomi, che corrispondono a indirizzi. Gli indirizzi sono espressi in modo relativo rispetto all'inizio del programma, permettendo così più semplici modifiche. Il programma, per essere eseguito, viene tradotto in linguaggio macchina da un programma traduttore detto **assemblatore**.

Alto livello:

1. **Linguaggi procedurali** Le operazioni disponibili sono ampie e non legate a quelle fornite dall'*hardware*. I dati sono indicati in modo totalmente indipendente dalla loro memorizzazione. Si tratta di linguaggi progettati affinchè la scrittura dei programmi sia semplice, elegante e, dunque, di facile comprensione e verifica. Per essere eseguito, un programma deve essere tradotto in linguaggio macchina da un programma traduttore detto **compilatore**, oppure deve essere interpretato (cioè eseguito da un altro programma, l'interprete). Linguaggi procedurali di rilievo sono (o sono stati) FORTRAN, COBOL, BASIC, Pascal, C. Tra i linguaggi procedurali alcuni sono detti orientati agli oggetti; tra questi ricordiamo C++ e Java.

2.

Oggi tutti i progetti industriali si realizzano in linguaggi ad alto (ed altissimo) livello.

La programmazione in linguaggio assembler è limitata alla realizzazione di semplici driver.

Da un punto di vista didattico - insegnare a programmare - i linguaggi ad alto livello sono evidentemente la scelta d'elezione:

1. permettono di concentrarsi sulla logica dell'algoritmo;
2. forniscono costrutti per descrivere in modo conciso tale logica;
3. mettono a disposizione strumenti per il test del programma.

Linguaggio di alto livello

A basso livello (livello macchina o assembler) un programma è una sequenza di istruzioni operative interrotta da istruzioni di salto:

```
"se si verifica una certa condizione, vai all'istruzione numero xx"
```

La scrittura di programmi complessi diviene difficile, perché il programma non rispetta la struttura della soluzione: i salti si incrociano l'uno con l'altro ed è impossibile seguire la logica della soluzione. Il risultato si chiama in gergo "programmazione a spaghetti".

Per risolvere un problema complesso:

1. lo si decompone in parti più semplici, possibilmente indipendenti;
2. si risolve ogni parte separatamente;

3. si combinano le varie soluzioni per ottenere la soluzione al problema originario.

I **linguaggi ad alto livello** mettono a disposizione strumenti linguistici per descrivere, all'interno del programma stesso, la struttura logica della decomposizione.

I principali strumenti sono:

1. strutture di controllo Invece del semplice salto dell'assembler, troviamo costrutti che gestiscono il flusso del controllo in modo strutturato;
2. strutture di modularizzazione Parti distinte della soluzione sono programmate come programmi distinti e indipendenti;
3. strutture dati Il linguaggio fornisce costrutti per definire e manipolare dati strutturati.

Prendiamo in considerazione queste tre caratteristiche in successione.

Costrutti di controlli strutturato

Solo alcune forme particolari di controllo (p.e. di iterazione) sono permesse nei moderni linguaggi ad alto livello. Le più rilevanti sono:

1. **iterazione determinata**: *for* i := <inizio> *to* <fine> *do* <corpo del *for*>
   <corpo del *for*> è eseguito <fine>-<inizio> volte, al variare della variabile i
2. **iterazione indeterminata**: *while* <condizione> *do* <corpo del *while*>
   <corpo del while> è eseguito fino a quando <condizione> non diventa vera
3. **condizionale**: *if* <condizione> *then* <ramo *then*> *else* <ramo *else*>
   se <condizione> è vera, si esegue <ramo then>; altrimenti si esegue <ramo else>
4. **eccezione**: *try* <comando> *catch* <gestore-errore>
   si esegue <comando>; se si verifica un errore (specificato in <gestore-errore>) si esegue <gestore-errore>

è chiaro che queste strutture di controllo si possono realizzare anche a basso livello.

Un linguaggio ad alto livello impone che tutti i programmi siano redatti usando solo queste strutture, che sono più semplici da analizzare della "programmazione a spaghetti".

Costrutti di modularizzazione

Per poter efficacemente decomporre la soluzione di un problema, i linguaggi ad alto livello permettono di risolvere una parte del problema in modo indipendente dalla soluzione (cioè dal programma) di altre parti. Per far ciò mettono a disposizione:

1. **procedure** e **funzioni**: Porzioni di programmi relativemente autosufficienti contraddistinte da un nome e da argomenti; codificano la soluzione di una parte del problema: per invocarli non è necessario conoscere i dettagli della soluzione, ma solo il nome e quali argomenti richiedono;
2. **parametri**: Argomenti delle procedure, mediante i quali è gestito il flusso delle informazioni dal programma principale alla procedura e viceversa;
3. **ambiente dinamico**: I nomi delle variabili e i loro spazi di memorizzazione sono gestiti al livello della singola procedura, senza necessità di aver nomi unici in tutto il programma.

Costrutti per la strutturazione dei dati

Il linguaggio fornisce in modo diretto costrutti per definire e manipolare dati strutturati.

1. nuovi tipi, oltre a dati numerici, si possono definire ed usare nel linguaggio nuovi tipi di dato:

1. enumerazioni;
2. insiemi;
3. matrici;
4. aggregati eterogenei (record);
5. ecc.

L'uso di questi tipi permette di rappresentare in modo naturale la logica del problema, invece di ricorrere a codifiche artificiali;

1. controllo automatico dei tipi La presenza di un ricco sistema di tipi permette di rilevare alcuni errori semantici, che si manifestano come errori di tipo. Per esempio:
   3+pippo deve essere sbagliato, non si possono sommare interi e stringhe.

Il controllo dei tipi è analogo al controllo dimensionale in fisica: se una formula che esprime una velocità non è uno spazio diviso un tempo, la formula deve essere sbagliata.

Il controllo può essere:

1. statico prima dell'esecuzione del programma;
2. dinamico durante l'esecuzione del programma.

In ogni caso è il sistema che rileva e segnala l'errore.