

La programmazione

Concetti
introduttivi

LINGUAGGIO MACCHINA

All'inizio, negli anni '40, l'unico metodo per programmare era il linguaggio macchina (ASSEMBLER), in altre parole il lavoro del programmatore era quello di settare ogni singolo bit a 1 o 0 su enormi computer che occupavano stanze intere e pesavano decine di tonnellate; si capisce che questo tipo di procedimento, non solo era faticoso, ma era riservato ad una cerchia ristretta di persone.

FORTRAN e ALGOL

Pochi anni dopo il 1950, con il progresso tecnologico e la riduzione delle dimensioni e dei costi dei calcolatori, nacquero due importanti linguaggi di programmazione, il FORTRAN (FORMula TRANslator), il cui utilizzo era ed è prettamente quello di svolgere in maniera automatica calcoli matematici e scientifici, e l'ALGOL (ALGOrithmic Language), altro linguaggio per applicazioni scientifiche sviluppato da Backus (l'inventore del FORTRAN) e da Naur, i quali misero a punto un metodo per rappresentare le regole dei vari linguaggi di programmazione che stavano nascendo.

COBOL

Nel 1960 venne presentato il COBOL (COMmon Business Oriented Language), ideato per applicazioni nei campi dell'amministrazione e del commercio, per l'organizzazione dei dati e la manipolazione dei file.

BASIC

Nel 1964 fa la sua comparsa il BASIC, il linguaggio di programmazione per i principianti, che ha come caratteristica fondamentale quella di essere molto semplice e, infatti, diventa in pochi anni uno dei linguaggi più utilizzati al mondo.

PASCAL

Intorno al 1970, però, Nikluis Wirth pensò bene di creare il PASCAL per andare incontro alle esigenze di apprendimento dei neo-programmatori, introducendo però la possibilità di creare programmi più leggeri e comprensibili di quelli sviluppati in basic. Si può affermare con certezza che Wirth ha centrato il suo obiettivo, considerando che ancora oggi il Pascal viene usato come linguaggio di apprendimento nelle scuole.

LINGUAGGIO C

Pochi anni più tardi fa la sua comparsa il C che si distingueva dai suoi predecessori per il fatto di essere molto versatile nella rappresentazione dei dati. Il C, infatti, ha delle solide basi per quanto riguarda la strutturazione dei dati, ma appare povero per la limitatezza degli strumenti a disposizione. Esso viene considerato "il linguaggio di più basso livello tra i linguaggi ad alto livello", per la sua potenza del tutto paragonabile al linguaggio macchina, mantenendo però sempre una buona facilità d'uso.

LINGUAGGIO C ++

Ma la vera rivoluzione si è avuta nel 1983 quando Bjarne Stroustrup inventò il C++ (o come era stato chiamato inizialmente "C con classi") che introduceva, sfruttando come base il C, la programmazione Orientata agli Oggetti (OO - Object Oriented) usando una nuova struttura, [la classe](#).

ALTRI LINGUAGGI

Altri linguaggi di programmazione da menzionare sono il LISP (1959), l'ADA (1970), lo SMALLTALK (1970) e il LOGO. Negli ultimi cinque anni, inoltre, si sono fatti strada linguaggi di programmazione come il JAVA, il JAVASCRIPT, linguaggi di scripting come l'ASP, PERL e il PHP; i "vecchi" linguaggi, come il Basic, il Pascal o il C++, sono diventati "VISUALI" e, di recente, ".NET". In questo modo si è aperta la strada ad una programmazione più intuitiva e veloce e particolarmente adatta all'implementazione in rete.

QUALE LINGUAGGIO SCEGLIERE?

I linguaggi di programmazione possono fare benissimo alcune cose e male altre. Per questo devono essere scelti opportunamente. I maggiori campi di utilizzo sono i seguenti:

- [Applicazioni Matematiche](#)
- [Prog. MultiPiattaforma](#)
- [Sistemi Operativi](#)
- [Prog. di Apprendimento](#)
- [Programmi Gestionali](#)
- [Creazione di Giochi](#)
- [Programmi per Windows](#)
- [Sviluppo Siti Dinamici](#)

Programmazione

Per formalizzare in maniera logica i concetti che vogliamo trasformare in programma vengono utilizzati [l'Algebra di Boole](#) (utile per le operazioni sui dati) ed i [Diagrammi di Flusso](#) (utili per rappresentare in maniera ordinata i nostri processi logici). La corretta scrittura del codice dipenderà dalla giusta rappresentazione del flusso delle istruzioni che useremo per risolvere il problema.

Programmazione

Un computer è una macchina costituita da un insieme di componenti elettroniche: il suo funzionamento dipende da un flusso continuo di segnali elettrici, sincronizzati da un clock, che codificano, in notazione binaria, le informazioni immagazzinate ed elaborate all'interno della macchina stessa.

Programmazione

Schematicamente le caratteristiche principali di un calcolatore sono:

- ❑ **velocità**: è rapidissimo nell'eseguire i compiti che gli vengono assegnati;
- ❑ **precisione**: è estremamente preciso nell'eseguire calcoli matematici e logici;
- ❑ **affidabilità**: svolge con rigore e puntualità i compiti assegnati dall'utente;
- ❑ **duttilità**: è in grado (se programmato correttamente) di svolgere lavori assai diversi.

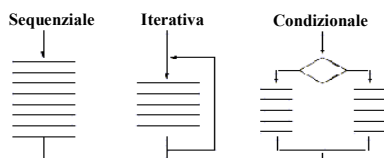
Programmazione

Il computer può essere **istruito** per svolgere compiti anche molto differenti tra loro. Queste istruzioni sono assegnate per mezzo di un **programma**, ossia tramite una sequenza ben definita di istruzioni che la macchina è in grado di tradurre in funzioni elementari facilmente eseguibili.

[Esempio di programma](#)

Programmazione strutturata

La **programmazione strutturata** rende più chiari e leggibili i programmi. Questo paradigma impone l'uso esclusivo di tre strutture di base per la costruzione di ogni algoritmo. Queste tre strutture possono essere combinate tra loro, giustapponendole o nidificandole una dentro l'altra.



Programma

Per poter essere eseguito da un elaboratore elettronico, ogni algoritmo deve essere codificato mediante le parole di un **linguaggio di programmazione**.

Un simile linguaggio è costituito da un insieme di "**Parole chiave**", "**Espressioni letterali**", "**Operatori e separatori**" e di regole sintattiche che ci consentono di tradurre in termini precisi e non ambigui i singoli passi che costituiscono un algoritmo.

Esempio di programma

L'unico linguaggio con cui il computer è in grado di operare è il linguaggio macchina. Sono necessari quindi appositi programmi che traducano le istruzioni dei programmi scritti con un linguaggio di programmazione. Questo tipo di programmi **traduttori** sono classificabili in due categorie principali, in base al loro modo di operare la traduzione:

- [compilatori](#)
- [interpreti](#)

Le variabili

L'elaboratore elettronico è in grado di memorizzare **i dati** su cui opera in apposite **celle di memoria**. Queste celle possono essere individuate in un linguaggio di programmazione di alto livello, mediante l'uso delle **variabili**. Queste si definiscono con un **nome** che le rappresenta e con cui si effettuano operazioni.

Le variabili, in programmazione, vengono intese come delle strutture in cui è possibile memorizzare dei dati. Il loro uso semplifica l'accesso diretto alle **locazioni di memoria**.

elle gi educational

Tipi di dati

Esistono diversi **tipi di dato** che possono essere memorizzati all'interno di un calcolatore elettronico. In particolare si definiscono i seguenti tipi:

- ❑ tipo Booleano;
- ❑ tipo Intero;
- ❑ tipo Reale;
- ❑ tipo Enumerazione;
- ❑ tipo Carattere.

elle gi educational

Struttura sequenziale

Il modo in cui strutturiamo un ragionamento, generalmente, segue la logica procedurale, cioè avendo delle operazioni da fare le eseguiamo una dopo l'altra in sequenza dalla prima all'ultima; anche nella programmazione avviene la stessa cosa. E' questa la **struttura sequenziale**.

A volte è necessario variare questa logica sequenziale introducendo strutture di tipo iterativo o condizionale. Per fare ciò si inseriscono **Istruzioni Condizionali**.

elle gi educational

Struttura condizionale

Il costrutto principale di una struttura condizionale è l'**IF**, il quale permette di porre una condizione ed eseguire delle scelte in base ai dati forniti in ingresso.

istruzione-if :

```
if ( espressione ) istruzione endif
if ( espressione ) istruzione else istruzione endif
```

L'espressione che compare dopo la parola chiave **if** è di tipo logico; se la condizione è vera viene eseguita l'istruzione seguente; nel secondo caso se la condizione risulta vera si esegue l'istruzione seguente, altrimenti si esegue l'istruzione subito dopo la parola chiave **else**.

elle gi educational

Struttura condizionale

Quando le scelte sono più d'una, è possibile **nidificare** le strutture **IF**. Un esempio è il seguente:

```
intero A = 50;
scrivi sullo schermo "Inserisci un numero tra 0 e 100";
intero B = -numero inserito da tastiera;
```

```
if (B < A) {scrivi sullo schermo "Il numero inserito è minore di cinquanta";}
else if (B > A) {scrivi sullo schermo "Il numero inserito è maggiore di cinquanta";}
else { //poiché A non è minore o maggiore deve essere uguale scrivi sullo schermo "Il numero inserito è cinquanta";}
endif
endif
```

elle gi educational

Struttura iterativa

La modalità comune di programmazione è quella sequenziale, ma esiste un'altra logica, meno comune: la **struttura iterativa**.

Il concetto fondamentale di questa struttura è quello di eseguire un'istruzione (o una serie di istruzioni) ciclicamente, fino a quando non sopraggiungono delle condizioni che fanno terminare tale computazione.

I tre costrutti iterativi comunemente usati sono il **WHILE**, il **DO** ed il **FOR**.

La CLASSE

Il concetto di classe è semplice, dividere l'interfaccia dal contenuto, ottenendo in questo modo tanti "moduli" interagenti tra loro attraverso le interfacce, permettendo così al programmatore di cambiare il contenuto di una classe senza per questo doversi preoccupare di controllare eventuale altro codice che richiami la classe.

Il C++ ha creato un nuovo modo di programmare, o meglio di progettare un programma, rendendo il codice scritto più chiaro e soprattutto "riutilizzabile".

Applicazioni Matematiche o di Ricerca

Per questo tipo di programmazione i linguaggi più adatti sono ancora il Fortran e l'Algol, non tanto perché non si possano creare linguaggi più versatili o potenti, ma perché dalla loro creazione è stato scritto ogni genere di programma matematico e quindi non è conveniente convertire tutto il codice già esistente in altro linguaggio di programmazione.



Sistemi Operativi

La programmazione di un sistema operativo richiede il controllo di tutte le risorse del computer e ciò è possibile solamente scrivendo il motore del sistema operativo nel linguaggio più vicino al linguaggio macchina, ovvero l'Assembler il quale è estremamente potente, ma anche estremamente difficile da imparare e, soprattutto, da usare bene; le parti restanti del sistema operativo (ad esempio l'interfaccia con l'utente) generalmente vengono scritte in C o in C++.



Programmi Gestionali

I programmi gestionali sono scritti in Visual Basic se devono girare su piattaforma *Windows*, e generalmente in C/C++ se sono implementati per sistemi operativi *Unix* o *Linux*.

Frutto di questi linguaggi sono anche i pacchetti applicativi "General Purpose" direttamente gestibili da utenti non programmatori. Un esempio è la suite Office della Microsoft.



Programmi per Windows

Per sviluppare programmi che girano sotto Windows esistono molti linguaggi, primo tra tutti il Visual Basic; altri sono il Visual C++, il Delphi (l'evoluzione visuale del Pascal) e il Java. Molti altri linguaggi permettono lo sviluppo di programmi per la piattaforma Windows, vista l'ampia diffusione di questo sistema operativo. Un esempio importante è costituito dai "programmi autore" per lo sviluppo di pubblicazioni multimediali (es. Flash Mx, Macromedia Director, Asymetrix Toolbook, ecc.).



Programmi MultiPiattaforma

Ad oggi l'unico vero linguaggio di programmazione MultiPiattaforma rimane il Java (prodotto dalla SUN), che pur con le sue pecche, permette di creare programmi che possono girare indistintamente su Windows, su Macintosh, su Linux (con molti problemi). Il punto di forza di Java rimane la Java Virtual Machine che, una volta installata su un qualsiasi sistema operativo, permette di eseguire i programmi Java senza problemi legati all'hardware o alla diversa impostazione dei sistemi operativi.



Programmi per l'apprendimento della Programmazione

Nei licei e nelle università per insegnare a programmare si utilizzano principalmente tre linguaggi, l'HTML, il Pascal e il C++. Attraverso di essi lo studente apprende i principali algoritmi di calcolo computazionale e la struttura di un programma.



Creazione di Giochi

Lo sviluppo di videogiochi avviene principalmente utilizzando il C++ e in alcune occasioni (per migliorare la velocità della grafica) anche l'Assembler. Bisogna dire però che possono essere creati semplici videogiochi anche utilizzando il Java ed il Flash Mx, anche se quest'ultimo non può essere considerato un vero linguaggio di programmazione.



Sviluppo Siti Dinamici

Per lo sviluppo di siti dinamici si possono utilizzare molti linguaggi, come l'Asp, il Perl, o il Php. L'Asp è nativo per la piattaforma Windows mentre gli altri due, oltre che per il Windows, sono adatti alle piattaforme Unix e Linux. Questi linguaggi sono da considerarsi soprattutto "linguaggi lato Server". Recentemente stanno emergendo anche Python, Zope (che utilizza il Python) e Jsp, un linguaggio di scripting che utilizza le librerie di Java.



Algebra di Boole

L'Algebra di Boole si applica a proposizioni di tipo logico a cui associamo il valore **1 (vero)** o il valore **0 (falso)**.

I principali operatori logici sono:

AND - Congiunzione

falso AND falso	risultato falso
falso AND vero	risultato falso
vero AND falso	risultato falso
vero AND vero	risultato vero

L'operatore AND restituisce un valore vero "se e solamente se gli altri due valori sono veri", questo vuol dire che in una successione di più operazioni AND se un valore è falso il risultato è falso.



Algebra di Boole

OR - Disgiunzione

falso OR falso	risultato falso
falso OR vero	risultato vero
vero OR falso	risultato vero
vero OR vero	risultato vero

L'OR restituisce un valore vero "se e solamente se almeno uno dei due valori risulta vero"; in una successione di più operazioni OR se un valore è vero anche il risultato è vero.



Algebra di Boole

NOT - Negazione

NOT falso	risultato vero
NOT vero	risultato falso

Il NOT si riduce ad una semplice operazione di negazione del valore acquisito.



I Diagrammi di Flusso

I diagrammi di flusso sono rappresentazioni grafiche che schematizzano un percorso logico come sequenza di operazioni elementari. Ciò consente l'individuazione di un algoritmo che risolva il nostro problema.

La rappresentazione degli elementi del flusso di operazioni fa uso di elementi grafici con significati specifici.



Simboli grafici

Punto di Partenza		l'azione che dà via al processo
Termine		la conclusione del processo
Documenti		sono i documenti di lavoro
Collaborazione		usato per indicare altre unità che potrebbero collaborare allo svolgimento del processo
Ingresso/Uscita		indica un'interazione con il mondo esterno, mostrando un risultato ottenuto
Assegnazioni		serve per assegnare dei valori o per definire delle costanti
Test e Confronti		utilizzato per suddividere il flusso secondo le casistiche, generalmente attende un dato fornito dall'utente per scegliere il percorso di esecuzione
Salto		freccia che indica la direzione attraverso la quale deve confluire un percorso



Esempio di formalizzazione

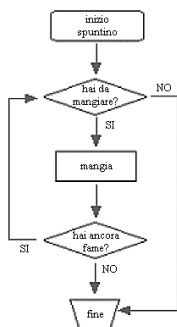
Volendo formalizzare un semplice ragionamento che verifichi la possibilità di fare uno spuntino, le operazioni da effettuare sono le seguenti:

- Inizia lo spuntino
- Hai a disposizione qualcosa da mangiare?
- Mangia
- Hai ancora fame?
- Fine dello spuntino

Così rappresentato l'algoritmo risulta di difficile comprensione; con il diagramma di flusso il problema è più chiaramente percepito nel suo percorso risolutivo.



Diagramma di flusso



La teoria dei diagrammi di flusso è comunque molto più complessa rispetto all'esempio presentato. Quanto esposto serve a farsi l'idea di come uno strumento apparentemente semplice, possa essere usato per formalizzare anche problemi di grande complessità.



Esempio di programma

Effettuare la media aritmetica tra 5 numeri forniti dall'utente. Il calcolatore deve leggere in input i cinque numeri, sommarli, memorizzare la somma nella sua memoria, dividere per cinque la somma totale e stampare il risultato.



Esempio di programma

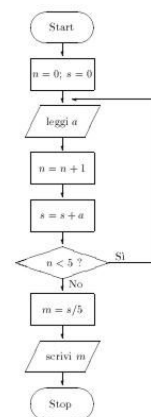
Una codifica dell'algoritmo risolutore potrebbe essere la seguente:

1. assegna ad n il valore 0
2. assegna ad s il valore 0
3. leggi un numero in input ed assegnalo ad a
4. incrementa di 1 il valore di n
5. somma a ed s ed assegna il risultato ad s
6. se n è minore di 5 vai al passo 3 altrimenti prosegui
7. dividi s per 5 ed assegna il risultato ad m
8. visualizza il valore di m
9. fermati



Esempio di programma

Rappresentazione con il diagramma di flusso



Esempio di programma

L'algoritmo, espresso in lingua italiana, per essere capito dal computer deve essere tradotto in *linguaggio macchina* (linguaggio basato sulla codifica binaria). Per fare ciò, sono stati progettati *linguaggi di programmazione ad alto livello*, ovvero un insieme di parole chiave e di regole sintattiche, facilmente traducibili (in modo automatico, mediante programmi chiamati *compilatori*). Essi permettono all'operatore di comunicare con la macchina in modo relativamente semplice.



Compilatori

I compilatori effettuano la traduzione delle righe del programma una volta per tutte e producono in output un programma in linguaggio macchina equivalente a quello fornito in input dal programmatore; il programma *compilato* potrà poi essere eseguito più volte senza necessità di traduzione; un programma compilato è molto più efficiente di un programma interpretato. Il C, il Fortran ed il Pascal sono linguaggi dotati di un compilatore.



Interpreti

I linguaggi di programmazione interpretati effettuano la traduzione di una istruzione di programma per volta e la eseguono immediatamente dopo averla tradotta; la traduzione avviene, quindi, durante l'esecuzione. Il vantaggio dell'uso di un interprete rispetto ad un compilatore sta nel fatto che il programma sorgente può essere modificato in ogni momento e poi inviato immediatamente in esecuzione, senza la necessità di dover eseguire ulteriori passaggi.



Parole chiave

Le parole chiave sono termini composti da caratteri alfanumerici, riservate al linguaggio di programmazione. Il creatore del linguaggio di programmazione assegna a queste parole una funzione specifica; il compito del programmatore è quello di usarle in maniera appropriata. Nella scrittura di un programma non è possibile definire funzioni o variabili utilizzando le parole chiave, pena segnalazione di errore o malfunzionamenti.



Espressioni letterali

Nei linguaggi di programmazione, le *espressioni letterali* vengono utilizzate per rappresentare valori costanti, stringhe, variabili intere o reali. Nel loro uso è necessario utilizzare specifiche regole sintattiche che impediscano ambiguità e consentano all'interprete del linguaggio la comprensione dell'algoritmo codificato per la soluzione del problema.



Operatori e separatori

In combinazione con le parole chiave vengono usati alcuni caratteri speciali, chiamati *operatori*, che controllano il flusso delle operazioni che devono essere eseguite. Gli operatori hanno delle proprietà che permettono di interpretare in modo univoco il significato di un'espressione. I *separatori* invece sono simboli di interpunzione che permettono di chiudere un'istruzione o di raggruppare degli elementi.



Tipo Booleano

Il tipo **booleano** è il tipo più semplice esistente, perché può assumere solamente i valori **vero** (o 1) e **falso** (o 0). Gli operatori su tali valori sono quelli dell'algebra di Boole.



Tipo Intero

Il tipo **intero** è costituito dai numeri interi compresi tra un limite inferiore ed un limite superiore; tali limiti dipendono dall'implementazione del compilatore e dal sottotipo (short, long, unsigned), ma generalmente sono costituiti da valori che vanno da -2^{N-1} a $+2^{N-1}$ dove N è il numero di bit utilizzati per rappresentare gli interi (generalmente 16 o 32).



Tipo Reale

Il tipo **reale** è formato da valori che appartengono ad un sottoinsieme dei numeri razionali; la rappresentazione è limitata prima di tutto dalle cifre presenti dopo la virgola; ad esempio, risulta impossibile rappresentare numeri periodici.

Inoltre esiste una limitazione inferiore e superiore, come quella del tipo intero, dipendente dal compilatore.



Tipo Enumerazione

Il tipo **enumerazione**, non presente in tutti i linguaggi di programmazione, serve a rappresentare costanti intere, predefinite dal programmatore, associate a informazioni non numeriche. L'esempio tipico può essere quello di associare un valore ai giorni della settimana per compiere delle operazioni.

Il vantaggio principale è quindi quello di poter eseguire operazioni matematiche su dati non numerici.



Tipo Carattere

Il tipo **carattere** (o stringa) rappresenta i caratteri che normalmente sono visibili sullo schermo o stampabili più alcuni caratteri speciali che dipendono dall'implementazione.

La codifica usata più spesso è quella ASCII in cui un carattere occupa un byte.



Costrutto While

L'istruzione **WHILE** viene schematizzata come segue:

istruzione-while :

while (*condizione*) *istruzione* **endwhile**

Con questa istruzione viene prima valutata la condizione racchiusa tra le parentesi tonde, se l'espressione risulta vera viene eseguita l'istruzione all'interno del while e il while viene ripetuto, altrimenti si esce dal ciclo del while e si procede con il resto del programma.



Costrutto Do

L'istruzione DO può essere considerata una variante dell'istruzione while ed è strutturato nella maniera seguente:

istruzione-do :

do *istruzione* **while** (*condizione*) ;

Prima di tutto viene eseguita l'istruzione racchiusa tra DO e WHILE (*quindi si esegue almeno una volta*), poi si verifica il risultato dell'espressione, se è vero si esegue di nuovo il DO, altrimenti si continua con l'esecuzione del resto del programma.



Costrutto For

L'istruzione FOR, schematizzata secondo la notazione di Backus-Naur, inizializza una variabile, pone una condizione (che deve essere vera o falsa) e poi modifica (incrementa o decrementa) la variabile iniziale.

istruzione-for :

for (*inizializzazione* ; *condizione* ; *incremento*)
istruzione

