MACROAREA **A**

Concetti base

A.1 FONDAMENTI DI INFORMATICA

1 Introduzione all'informatica
2 Hardware e software
3 Ambiente operativo

A.2 RETI E SICUREZZA

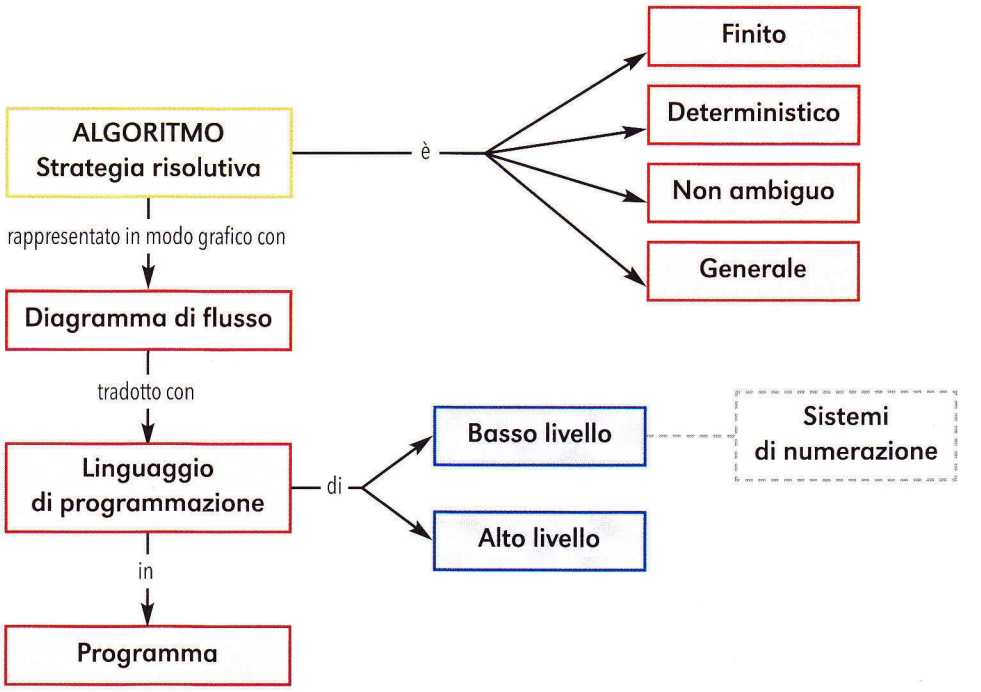
1 Reti, Internet e servizi
2 Sicurezza informatica

A.1 FONDAMENTI DI INFORMATICA

Uda 1 INTRODUZIONE ALL'INFORMATICA

OBIETTIVI UNITÀ

- Conoscere i concetti di algoritmo, diagramma di flusso, programma.
- Distinguere i diversi linguaggi di programmazione.
- Elencare i sistemi di numerazione binario, decimale ed esadecimale.
- Conoscere come avviene la rappresentazione digitale dei dati.



SEGUIMI!

L'**informatica** (dal francese *Information Automatique*) è la **scienza** che si occupa dei sistemi per elaborare e gestire in modo automatico le informazioni.

Tutte le volte in cui si deve risolvere un problema o soddisfare una necessità occorre:

- individuare gli elementi che si hanno a disposizione (**dati di input**);
- stabilire l'obiettivo che si vuole raggiungere (**dati di output**);
- precisare la **procedura** che si intende seguire per ottenere l'**obiettivo** prefissato.

Tante sono le situazioni che, nella vita quotidiana, richiedono l'esecuzione di una **procedura**, cioè il compimento di un una **sequenza precisa, ordinata e finita di azioni** che conducono all'obiettivo fissato. Tale sequenza, in informatica, viene descritta attraverso l'**algoritmo**.

Per comprendere meglio questo concetto, esaminiamo l'esempio di una famiglia che ha acquistato la macchina per fare il pane (*Figura 1*). Avremo, in questo caso:

- come **input**: farina, lievito, acqua, sale, olio;
- come **algoritmo risolutivo**: programma di impasto e cottura scelto (istruzioni che conducono al risultato fissato);
- come **output**: pane.

Figura 1

L'esempio può essere trasferito in ambito informatico e può aiutarci a comprendere l'attività svolta da un PC per soddisfare le nostre esigenze quotidiane (*Figura 2*).

Figura 2

Concetto di algoritmo

L'algoritmo è la **strategia risolutiva** di un problema, costituito da una **sequenza finita di operazioni** (dette anche **passi** o **istruzioni**) che consente di risolvere tutti i problemi di una determinata "classe" e produrre il risultato stabilito.

L' algoritmo, in quanto tale, deve essere:

- **finito**, ovvero essere costituito da un numero finito di passi eseguiti un numero finito di volte;
- **deterministico**, cioè, a partire dagli stessi dati di input, deve produrre i medesimi risultati;
- **non ambiguo**, in quanto le operazioni che lo costituiscono devono poter essere interpretate in modo univoco anche da esecutori diversi;
- **generale**, ovvero deve fornire la soluzione per tutti i problemi che appartengono alla medesima classe.

SEGUIMI!

I dati che interessano l'algoritmo possono essere:
- **costanti**, quando nello svolgimento dell'algoritmo mantengono sempre lo stesso valore (si tratta in genere di numeri);
- **variabili**, quando durante l'elaborazione possono assumere valori diversi (vengono in generale rappresentate da lettere, a ognuna delle quali sarà assegnata una determinata variabile).

Diagrammi di flusso

Per descrivere in **forma algoritmica** la **procedura risolutiva** di un problema è necessario rendere visivamente comprensibile la **successione logica** delle **istruzioni** di cui si compone l'algoritmo procedendo con la sua **rappresentazione grafica** mediante la costruzione di **diagrammi di flusso**, definiti anche diagrammi a blocchi, o *flow-charts*. Tali diagrammi costituiscono lo strumento per **presentare il ragionamento** che condurrà al risultato atteso.
I **blocchi** di cui si compongono sono figure geometriche nelle quali sono inserite le **operazioni elementari**. Tali blocchi sono collegati tra loro da linee di giunzione, orientate e disposte per stabilire la sequenza in cui si desidera che le operazioni vengano svolte, con possibilità di collegamenti diversi per risolvere le molteplici situazioni che si presentano e che dovranno essere soddisfatte dall'algoritmo.
In algoritmi realizzati per la soluzione di problemi matematici, le istruzioni si sviluppano mediante **costanti e variabili** legate da **operatori aritmetici**, per esempio **+**, **−**, * ecc., il cui risultato sarà assegnato a una variabile.

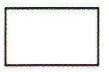
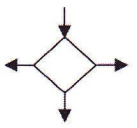
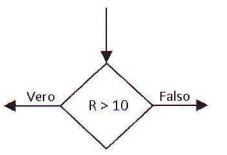
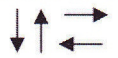
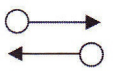
ESEMPIO 1

Di seguito mostriamo un esempio di istruzioni espresse con costanti e variabili.

A = B − 10 *significa*

Sottrai al valore della **variabile B** il valore della **costante 10** e **assegna** il risultato alla **variabile A**.

Significato dei blocchi dei diagrammi

BLOCCO	SIGNIFICATO
Inizio 	Serve per indicare il punto di partenza dell'algoritmo e al suo interno possono apparire le scritte **INIZIO** o **START**: in un diagramma di flusso ne è presente solo uno.
Ingresso/ Uscita 	Viene utilizzato per le operazioni di ingresso o scrittura dei dati (**input**) e di uscita o lettura dei risultati (**output**). Ha una sola linea di giunzione in entrata e in uscita.
Elaborazione/ Azione 	Serve per le operazioni generiche (operazioni di calcolo) o di assegnazione e al suo interno viene descritta in forma simbolica o letterale l'**operazione da eseguire**. Ha una sola linea di giunzione in entrata e in uscita.
Decisione/ Domanda/ Confronto 	Serve per le operazioni di **confronto**, **controllo** e di **scelta** tra alternative per percorrere sequenze di istruzioni diverse in funzione del verificarsi o meno di certe situazioni; può essere a un solo ingresso e a due o più uscite. Al suo interno si inserisce l'espressione da considerare, il cui risultato viene confrontato con variabili e costanti attraverso gli operatori = uguale, > maggiore, >= maggiore o uguale, < minore, <= minore o uguale, <> diverso, **V** (vero), **F** (falso), indicati sulle linee di giunzione (in questo caso definite di uscita). Nell'esempio proposto si dovrà procedere con le operazioni in un senso oppure nell'altro in base al fatto al fatto che la condizione posta risulti vera oppure falsa. 
Linee di giunzione/ Direzione del flusso 	La **freccia** posta al termine di ogni blocco o segmento **collega al passo** logico **successivo** da prendere in considerazione e quindi indica il flusso di esecuzione delle operazioni.
Fine 	Serve per indicare la fine dell'algoritmo e nel suo interno possono apparire le scritte **FINE** o **STOP**. In genere in un algoritmo ne è presente uno solo.
Connettori 	Individuano dei punti dai quali si può fare riferimento ad altre parti. Per esempio, il connettivo ①→ trasferisce il proseguimento del diagramma dal punto in cui si trova al punto dove compare il connettore →① contenente lo stesso numero.

Gli algoritmi vengono utilizzati per costruire diagrammi, o flow chart, e per la soluzione di problemi di calcolo.

ESEMPIO 2

Consideriamo il seguente problema: dato in input un numero, calcolarne il cubo se è maggiore di 0, il quadrato in caso contrario.

VARIABILI UTILIZZATE

- **NUM**, che rappresenta il numero (reale) in input;
- **RIS**, che rappresenta il risultato (reale) in output.

Se osserviamo l'algoritmo risolutivo di *Figura 3*, vediamo che innanzitutto è necessario prendere in input un numero, effettuarne il confronto del valore con zero, quindi:

- se il numero è maggiore di zero, dovrà essere moltiplicato tre volte per se stesso;
- se il numero è minore di zero, dovrà essere moltiplicato due volte per se stesso.

Il risultato sarà sicuramente un numero positivo, oppure sarà uguale a zero se il numero di partenza è 0.

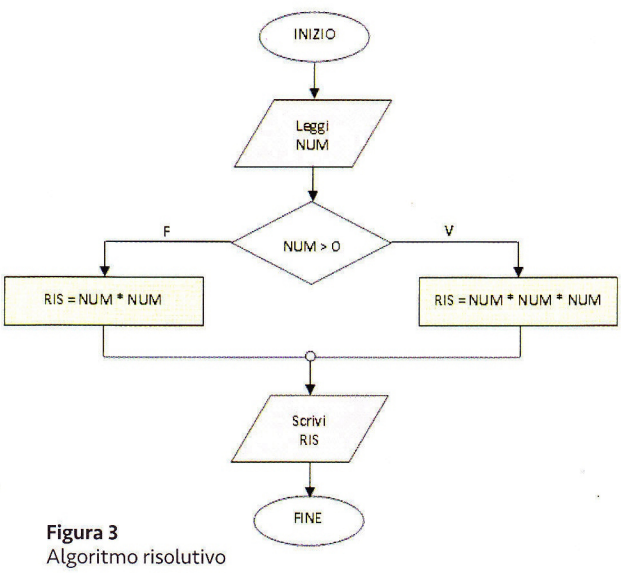


Figura 3
Algoritmo risolutivo

I connettivi logici AND e OR

SEGUIMI!

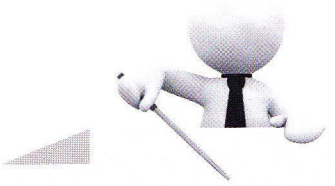
I valori **vero** o **falso** sono definiti **valori logici** o **valori booleani**.

Come si può notare dalla rappresentazione dell'algoritmo, l'elaboratore è programmato per eseguire determinate funzioni che si basano sulla verità o sulla falsità di certe proposizioni semplici, come per esempio "Il Po è un fiume" (proposizione semplice vera), oppure "2 + 2 = 3" (proposizione semplice falsa). La verità o la falsità di una proposizione semplice viene indicata con i termini "valore di verità", nel senso che una proposizione può essere **vera** o **falsa**, ma non entrambe le cose.
Le proposizioni semplici possono tuttavia essere combinate tra loro mediante i connettivi logici **AND** e **OR** in modo da dare origine a una nuova proposizione, detta "composta", il cui valore di verità è determinato dal valore di verità delle sue proposizioni semplici e dai connettivi logici che le collegano.

Area digitale

- La negazione logica NOT

Un connettivo, o operatore logico, **istituisce fra due proposizioni a e b una relazione che dà origine a una terza proposizione c** che sarà vera o falsa in base ai valori delle due proposizioni **a** e **b** e al tipo di connettivo utilizzato.



Di seguito prendiamo in considerazione le proposizioni semplici **a** e **b** collegate ai connettivi logici **AND** e **OR**, mostrando le relative **tavole di verità**.

Connettivo AND (congiunzione)

Date le proposizioni semplici **a** e **b** collegate dal connettivo **AND**, il risultato della congiunzione **a AND b** darà origine a una proposizione composta che sarà:

- **vera**, solo se **a** e **b** sono entrambe vere;
- **falsa**, quando almeno una delle due è falsa.

Tavola di verità della proposizione composta a AND b

a	b	a AND b
v	v	v
v	f	f
f	v	f
f	f	f

ESEMPIO

a (prima proposizione)	Connettivo AND	b (seconda proposizione)	Proposizione composta
Il Po è un fiume	e	2+2=4	Vero
Il Po è un fiume	e	2+2=3	Falso
Il Po è un lago	e	2+2=4	Falso
Il Po è un lago	e	2+2=3	Falso

Connettivo OR (disgiunzione)

Date le proposizioni semplici **a** e **b** collegate dal connettivo **OR**, il risultato della disgiunzione **a OR b** darà origine a una proposizione composta che sarà:

- **falsa**, solo se se **a** e **b** sono entrambe false;
- **vera**, quando almeno una delle due è vera.

Tavola di verità della proposizione composta a OR b

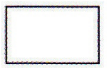
a	b	a OR b
v	v	v
v	f	v
f	v	v
f	f	f

ESEMPIO

a (prima proposizione)	Connettivo OR	b (seconda proposizione)	Proposizione composta
Il Po è un fiume	oppure	2+2=4	Vero
Il Po è un fiume	oppure	2+2=3	Vero
Il Po è un lago	oppure	2+2=4	Vero
Il Po è un lago	oppure	2+2=3	Falso

AL VOLO!

Osserva la figura. Si tratta di un blocco di:



○ elaborazione
○ ingresso

Il linguaggio naturale e i linguaggi di programmazione

In generale, quando si parla di **linguaggio**, ci si riferisce:

- al **lessico**, ossia all'insieme delle parole, ciascuna con il suo significato, disponibili per esempio in un dizionario;
- alla **sintassi**, costituita dalle regole da seguire per costruire una frase in modo corretto, affinché possa essere compresa.

Il **linguaggio naturale** è quello comunemente utilizzato per comunicare con gli altri e consente ricchezza espressiva; in alcuni casi, tuttavia, può dare adito ad ambiguità e talvolta anche a ridondanza. Vediamo un esempio.

ESEMPIO 3

Consideriamo la seguente frase: "**La bella pesca**".
Essa può essere interpretata in due modi:
1 "**una bella ragazza sta pescando**";
2 "**un bel frutto**".

L'esempio appena esaminato ci fa comprendere che il linguaggio naturale non può essere utilizzato per "istruire" l'elaboratore su ciò che deve compiere, in quanto le caratteristiche che presenta non sono adatte alla sua logica: **un elaboratore non può gestire informazioni ambigue**.

L'**algoritmo**, invece, per le sue caratteristiche, è **idoneo a istruire un elaboratore**. Tuttavia, le istruzioni di cui si compone, devono essere tradotte, mediante un **linguaggio di programmazione** (costituito da un **alfabeto** e da un insieme di **regole sintattiche** per l'uso corretto delle parole del linguaggio), in un **programma**.

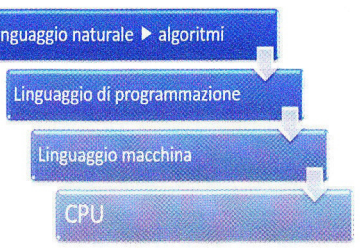


Figura 4

Il **linguaggio di programmazione** è un linguaggio **intermedio** fra **linguaggio macchina** (direttamente comprensibile dall'elaboratore) e il linguaggio che descrive gli algoritmi (*Figura 4*).

Esso viene definito anche **linguaggio di alto livello** in quanto è costituito da una sintassi che non è dipendente dal funzionamento specifico di una certa CPU: di conseguenza, consente di scrivere un programma indipendentemente dall'elaboratore su cui sarà eseguito.

I linguaggi di programmazione di alto livello e il linguaggio macchina

SEGUIMI!

Le operazioni elementari eseguite dalla CPU sono rappresentate come sequenze di cifre binarie.
Di conseguenza, a questo livello, un programma corrisponde a una serie di **istruzioni espresse in codice binario, direttamente eseguibili dalla CPU**.

I linguaggi di programmazione di alto livello tendono ad assomigliare ai linguaggi naturali proprio per rendere più facile la programmazione del computer. Il computer, invece, comprende solo il **linguaggio macchina (o linguaggio di basso livello)**, direttamente comprensibile dalla CPU, costituito da una sintassi limitatissima e molto rigida, composta da sequenze di **cifre binarie** 1 e 0.

I primi computer si basavano esclusivamente sul linguaggio macchina: il programmatore utilizzava una tabella che serviva per tradurre i comandi nella serie corrispondente di cifre binarie da inserire; era sufficiente sbagliare una cifra per mettere in crisi tutto il sistema. Inoltre il linguaggio di basso livello, direttamente compreso ed eseguito dal processore, è strettamente collegato alla sua struttura fisica, con la conseguenza che il medesimo programma può non funzionare in microprocessori diversi.

SEGUIMI!

Il linguaggio **Assembler** è un **linguaggio di programmazione di basso livello** che ha la stessa struttura del linguaggio macchina ed è strettamente legato alle caratteristiche del computer per il quale è stato definito. È complicato da memorizzare in quanto vi è una precisa corrispondenza tra le istruzioni simboliche del linguaggio Assembler e le istruzioni definite in linguaggio macchina.

Proprio per ovviare a questi inconvenienti sono stati sviluppati i **linguaggi di programmazione di alto livello**, in cui le istruzioni non sono più indicate da sequenze di cifre binarie, ma da nomi simbolici, più facili da riconoscere, memorizzare e utilizzare da parte del programmatore.

Tuttavia, poiché l'elaboratore riesce ad interpretare solo istruzioni formulate in linguaggio macchina nelle sequenze di 1 e 0, sarà necessario tradurre i programmi scritti in linguaggio di alto livello in linguaggio macchina.
Questa operazione di traduzione può essere eseguita dai programmi di seguito descritti.

- **Programmi compilatori**: traducono l'intero programma scritto in linguaggio di alto livello (per esempio, in Pascal o in C++) nella corrispondente copia in linguaggio macchina; tutte le istruzioni vengono controllate nel lessico e nella sintassi, tradotte e trasformate in un file eseguibile (con estensione **.exe** nei sistemi Windows) che diventa indipendente dal programma scritto in linguaggio di alto livello, potendo così essere eseguito senza il programma compilatore.
- **Programmi interpreti**: "leggono" riga per riga le istruzioni scritte in linguaggio di alto livello, ne controllano il lessico e la sintassi e le traducono in linguaggio macchina per farle eseguire direttamente dall'unità centrale di elaborazione. In questo caso non viene prodotta una copia del programma in linguaggio macchina, ma ogni istruzione viene di volta in volta tradotta e poi fatta eseguire.

AL VOLO!

Il linguaggio macchina è un linguaggio:

○ di alto livello
○ di basso livello

Per rispondere alle diverse esigenze sono stati sviluppati diversi linguaggi di programmazione di alto livello, alcuni dei quali sono indicati di seguito:

- **SCRATCH**, ambiente visuale particolarmente utilizzato in ambito scolastico per progetti pedagogici;
- **PYTHON**, orientato agli oggetti, è adatto per lo sviluppo software;
- **C**, utilizzato per applicazioni di tipo ingegneristico;
- **JAVA**, progettato per applicazioni su Internet;
- **PASCAL**, particolarmente adatto per la didattica della programmazione;
- **BASIC**, inizialmente progettato per applicazioni scientifiche e didattiche, viene attualmente utilizzato per ogni tipo di problema. Grazie alla facilità con cui viene assimilato e alla grande diffusione dei PC, oggi viene usato nella versione **VISUAL BASIC**.

Area digitale

- Utilizzo di pseudo linguaggi

ESEMPIO 4

Di seguito rappresentiamo in linguaggio PASCAL e in linguaggio VISUAL BASIC il diagramma proposto a pag. 4.

PASCAL

```
program NUMERO;
var
   NUM : Real;
   RIS : Real;
begin
   Writeln ('DAMMI IL NUMERO');
     Readln(NUM);
     if NUM > 0
        then
           RIS:= NUM * NUM * NUM
        else
           RIS:= NUM * NUM;
   Writeln ('IL RISULTATO E'': ', RIS:10:2);
end.
```

VISUAL BASIC

```
Sub NUMERO()

   Dim NUM     As Single              ' numero in input
   Dim RIS     As Single              ' risultato

   NUM = Val(InputBox("Dammi il numero"))
   If NUM > 0 Then
      RIS = NUM * NUM * NUM
   Else
      RIS = NUM * NUM
   End If
   MsgBox ("il risultato è "  & Str(RIS))
End Sub
```