

# Algoritmi

Il termine algoritmo proviene dalla matematica e deriva dal nome di un algebrista arabo del IX secolo di nome **Al-Khuwarizmi** e sta ad indicare un procedimento basato su un numero finito operazioni che specificano in modo rigido e meccanico ogni singolo passo del procedimento.

*Per **algoritmo** si intende la descrizione di insieme finito di istruzioni che devono essere eseguite per portare a termine un dato compito e per raggiungere un risultato finale definito in precedenza*

Esempi di algoritmi possono essere le istruzioni per l'uso di un elettrodomestico, una ricetta di cucina, le regole per la risoluzione di una equazione, ecc.

In ogni caso si può facilmente dedurre che un algoritmo presuppone la presenza di un **esecutore** in grado di interpretare ed eseguire correttamente le istruzioni.

Prima di arrivare a definire come costruire un algoritmo definiamo le proprietà che devono avere le istruzioni:

- Ogni istruzione deve essere **concretamente realizzabile** dall'esecutore (ad esempio l'istruzione "calcola la ventesima cifra dopo la virgola del numero Pi greco" non può essere eseguita da un esecutore quale una calcolatrice con sole 8 cifre)
- Le istruzioni devono essere **precise, non ambigue**. L'esecutore deve essere in grado di interpretare una istruzione in modo univoco
- Le istruzioni devono poter essere **eseguite in un tempo finito (limitato)**. Ad esempio l'istruzione "calcola tutte le cifre decimali del numero Pi-greco", essendo tale numero irrazionale, richiederà per la sua esecuzione un tempo infinito

- Le istruzioni devono produrre un risultato **osservabile**. Dobbiamo quindi essere in grado di verificare quale è il risultato dell'esecuzione.
- Le istruzioni devono essere **deterministiche**. Una stessa istruzione, a partire dalle stesse condizioni iniziali, deve fornire sempre lo stesso risultato.
- Le istruzioni devono essere **elementari (atomiche)**. Una istruzione non deve essere ulteriormente scomponibile in altre istruzioni. Osserviamo che una stessa istruzione può essere elementare oppure no a seconda dell'esecutore: l'istruzione "calcola la radice quadrata di un numero intero con approssimazione di 4 cifre" non è elementare se l'esecutore è una calcolatrice dotata delle sole 4 operazioni somma, sottrazione, prodotto e divisione.

Vediamo ora i criteri per la definizione di un algoritmo:

- La lista delle istruzioni che definiscono un algoritmo deve essere **finita**
- L'algoritmo deve essere **esaustivo**. Quando si progetta un algoritmo bisogna considerare tutti i possibili casi che si possono presentare e indicare la soluzione per ognuno di essi.
- L'algoritmo deve essere **riproducibile**. Ogni esecuzione dello stesso algoritmo con gli stessi dati iniziali deve fornire gli stessi risultati.

Vediamo ora un esempio: una segretaria deve contattare telefonicamente una serie di clienti a cui deve lasciare un messaggio.

### **Descrizione del problema**

La segretaria deve procurarsi la lista dei clienti da contattare ed il numero di numero di telefono di ciascuno di essi.

Per ogni cliente dell'elenco la segretaria dovrà comporre il numero, e nel caso in cui il numero risultasse occupato scriverà di fianco al numero nella lista "occupato". Se dovesse rispondere il cliente la segretaria annoterà "ok".

(Abbiamo volutamente ristretto i casi possibili a due alternative: l'esempio infatti sarebbe diventato troppo complesso).

**Dati necessari** (Dati di input)

Lista dei clienti da avvertire

Messaggio da comunicare

**Risultati** (Dati di output)

Lista dei clienti con l'indicazione, per ciascuno di essi, dell'esito della telefonata.

Algoritmo Risolutivo

**Ripeti**

Leggi il numero telefonico di un cliente

Componi il numero

**Se** numero occupato **allora**

    Annota "numero occupato"

**Altrimenti**

    Lascia messaggio

    Annota "ok"

**Finchè** elenco completato

L'algoritmo che abbiamo appena descritto ovviamente non rispetta alcune delle proprietà enunciate, poiché lo scopo dell'esempio è quello di individuare le **strutture** tramite cui organizzare le istruzioni.

- 1) Le istruzioni devono essere eseguite secondo la **sequenza** impostata: se nell'esempio si invertissero le istruzioni *Leggi il numero telefonico di un cliente* con *Componi il numero* l'algoritmo non avrebbe senso. L'algoritmo deve quindi essere eseguito dall'esecutore seguendo l'ordine stabilito.
- 2) Subito dopo aver composto il numero, ci si possono presentare due casi (si ricorda ancora una volta che le ipotesi sono semplificate) e, a seconda che il numero sia occupato oppure no si l'algoritmo prevede che si esegua l'istruzione *Annota "numero occupato"* **oppure** *Lascia messaggio - Annota "ok"*. La verifica della condizione *numero occupato* **seleziona** l'istruzione seguente da eseguire.
- 3) Nell'esempio una serie di istruzioni vengono ripetute un certo numero di volte; in particolare la **ripetizione** è controllata dalla **verifica di una condizione** che viene controllata ogni volta che la sequenza da ripetere è portata a termine.

Questi aspetti che abbiamo analizzato sono i tre concetti che sono alla base della costruzione di un algoritmo:

- **sequenza**
- **selezione**
- **iterazione**

Questi sono i concetti che sono alla base dei linguaggi di programmazione **imperativi**, famiglia di cui fa parte il linguaggio VB.NET.

Se ora analizziamo come siamo arrivati dal problema alla stesura dell'algoritmo potremo distinguere diverse fasi:

- 1) Individuazione dei dati iniziali sui quali basare la soluzione. Questa serie di dati viene chiamata **dati in ingresso** o **dati di input**.

- 2) Individuazione dei risultati che si vogliono ottenere. Questa serie di dati viene chiamata **dati di uscita** o **dati di output**
- 3) Individuazione delle risorse\* logiche e fisiche necessarie
- 4) Individuazione delle soluzioni, ovvero il procedimento da seguire per arrivare a definire l'algoritmo

Questi quattro punti costituiscono l'**analisi del problema**.

Terminate le fasi di analisi del problema, si deve definire l'algoritmo. A questo scopo si potrebbero utilizzare vari metodi ben codificati quali la **pseudocodifica** o i **diagrammi a blocchi**, ma vedremo che nel caso specifico di un linguaggio quale VB.NET, date le sue specifiche, potremo comunque saltare questa fase e passare direttamente alla stesura del codice.

---

\* Per risorse logiche si intendono le strutture dati, funzioni di libreria, ecc. mentre per risorse fisiche si intendono le specifiche hardware richieste.