

Gli algoritmi e i programmi

Mattia Cozzi
cozzimattia@gmail.com

a.s. 2024/2025



Contenuti

Introduzione

Pseudocodifica

Diagrammi

Strutture

Linguaggi

Programmi

Modello del problema

Il modello del problema è una **rappresentazione schematica** di un particolare aspetto della realtà.

Vengono individuate:

- le entità, cioè gli oggetti importanti ai fini della descrizione;
- le proprietà delle entità;
- le variabili, dati che possono variare;
- le costanti, dati che non cambiano.

Definizione di algoritmo

Il termine viene dall'algebrista persiano del IX secolo Muhammad ibn Musa **al-Khuwarizmi**.



Definizione

Un algoritmo è la descrizione di un insieme finito di istruzioni che devono essere eseguite per portare a termine un dato compito.

Esempi: istruzioni di montaggio di un mobile, ricetta di cucina, somme in colonna.

Esempio: la sequenza di Fibonacci

La famosissima sequenza di Fibonacci è costituita da numeri che vengono ottenuti sommando i due numeri precedenti della sequenza stessa. I primi due numeri sono 1:

1 1 2 3 5 8 13 21 ...

Possiamo operare con il seguente algoritmo:

- inizia con 1 e 1;
- per ottenere l' n -esimo numero F_n della sequenza, calcola $F_{n-1} + F_{n-2}$.

Il quadrato magico

Un quadrato magico è un quadrato con n numeri per lato, in cui la somma delle cifre di qualsiasi riga, colonna o diagonale fornisce lo stesso risultato.

2	7	6	→15	
9	5	1	→15	
4	3	8	→15	
↙15	↓15	↓15	↓15	↘15

Quadrato magico 3×3

Algoritmo per un quadrato magico, n dispari (1)

Per compilare un quadrato magico con lato dispari, possiamo usare il seguente algoritmo (esempio con $n = 5$).

1. Scrivere 1 nella riga superiore, al centro.

$$\begin{bmatrix} ? & ? & 1 & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix}$$

Algoritmo per un quadrato magico, n dispari (2)

2. Spostarsi a destra di una colonna e in su di una riga (ripartire dal basso se si è nella riga più alta, e da sinistra se si è già all'estrema destra) e scrivere il numero intero successivo.

$$\begin{bmatrix} ? & ? & 1 & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & 2 & ? \end{bmatrix}$$

3. Ripetere l'operazione precedente. Se la casella di destinazione è già occupata, scrivere il nuovo numero nella posizione immediatamente sotto a quella di partenza.

Algoritmo per un quadrato magico, n dispari (3)

$$\begin{bmatrix} ? & ? & 1 & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & 3 \\ ? & ? & ? & 2 & ? \end{bmatrix}$$

$$\begin{bmatrix} ? & ? & 1 & ? & ? \\ ? & 5 & ? & ? & ? \\ 4 & 6 & ? & ? & ? \\ ? & ? & ? & ? & 3 \\ ? & ? & ? & 2 & ? \end{bmatrix}$$

Algoritmo per un quadrato magico, n dispari (4)

$$\begin{bmatrix} ? & ? & 1 & 8 & ? \\ ? & 5 & 7 & ? & ? \\ 4 & 6 & ? & ? & ? \\ 10 & ? & ? & ? & 3 \\ 11 & ? & ? & 2 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

Pseudocodifica

È la descrizione di un algoritmo utilizzando il linguaggio comune secondo una serie di **regole rigorose** e un **vocabolario ristretto**.

Caratteristiche:

- Un algoritmo viene aperto e chiuso dalle parole **inizio** e **fine**.
- Operazioni di *input*: immetti, leggi, acquisisci, read.
- Operazioni di *output*: scrivi, mostra, comunica, write.

Operatori

Nel linguaggio di pseudocodifica possiamo utilizzare diversi operatori:

- assegnazione di un valore ad una variabile:

```
    assegna x = 9;  
    calcola y = x + 3;
```

- operatori matematici;
- operatori di confronto;
- operatori logici (AND, OR, NOT, XOR).

Parole chiave

In pseudocodifica si usano alcune parole speciali che permettono di **strutturare logicamente l'algoritmo**.

- se;
- allora;
- altrimenti;
- fine se;
- esegui;
- finché;
- mentre;
- ripeti.

Esempio 1

Algoritmo in pseudocodifica per il calcolo dell'area di un triangolo:

```
1  inizio
2      immetti base;
3      immetti altezza;
4      calcola area = .5 * base * altezza;
5      scrivi area;
6  fine
```

Esempio 2

Algoritmo in pseudocodifica per salutare in base all'ora del giorno:

```
1  inizio
2    acquisisci ora;
3    se ora < 12:00:
4      allora
5        scrivi 'Buongiorno';
6    se ora < 18:00:
7      allora
8        scrivi 'Buon pomeriggio';
9    altrimenti:
10     scrivi 'Buonasera';
11   fine se;
12 fine
```

Diagrammi a blocchi (*flowchart*)

I diagrammi a blocchi permettono di **rappresentare graficamente l'algoritmo**.

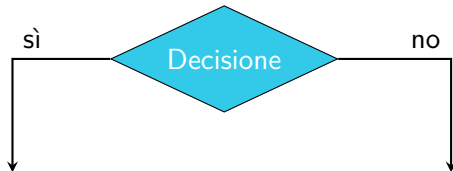
In questi schemi, blocchi di forme diverse hanno significati diversi.



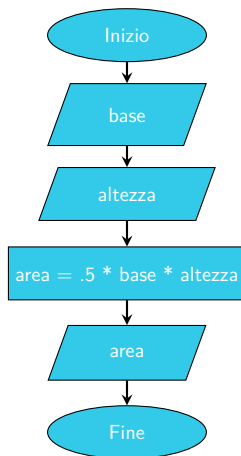
Tipi di blocchi

Elaborazione

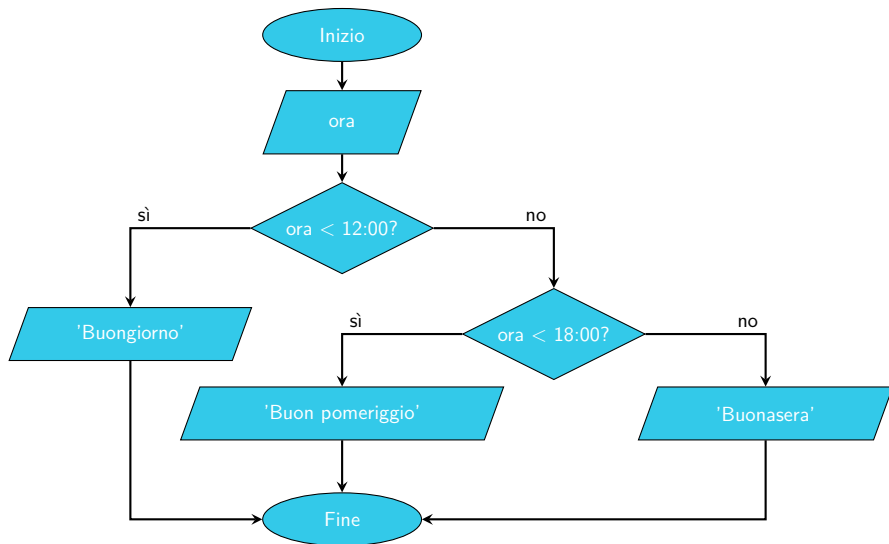
Input/Output



Esempio 1 a blocchi



Esempio 2 a blocchi



Strutture di controllo

Le istruzioni di un algoritmo possono:

- essere organizzate in **sequenza**;
- presentare delle **alternative** (struttura condizionale);
- essere **ripetute** un certo numero di volte o finché si verifica una certa condizione (struttura iterativa).

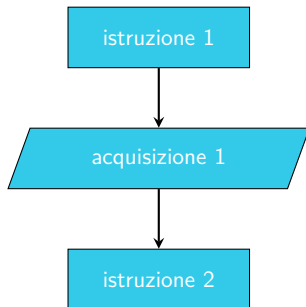
Ogni algoritmo può essere scritto con una combinazione di queste tre strutture fondamentali.

Sequenza

Pseudocodifica

```
1 istruzione 1;  
2 acquisizione 1;  
3 istruzione 2;
```

Blocchi

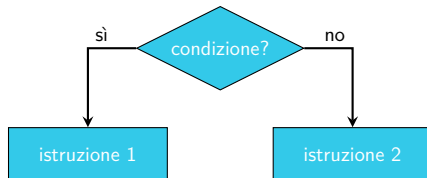


Struttura condizionale

Pseudocodifica

```
1  se condizione:  
2  allora  
3      istruzione 1;  
4  altrimenti  
5      istruzione 2;  
6  fine se
```

Blocchi

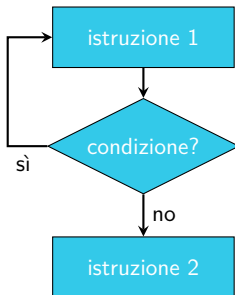


Iterazione

Pseudocodifica

```
1 esegui
2   istruzione 1;
3   ripeti mentre condizione;
4   istruzione 2;
```

Blocchi

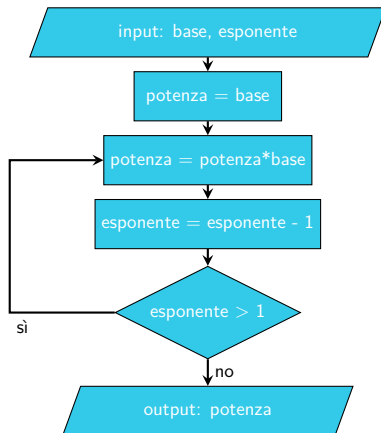


Esempio: calcolo di una potenza

Pseudocodifica

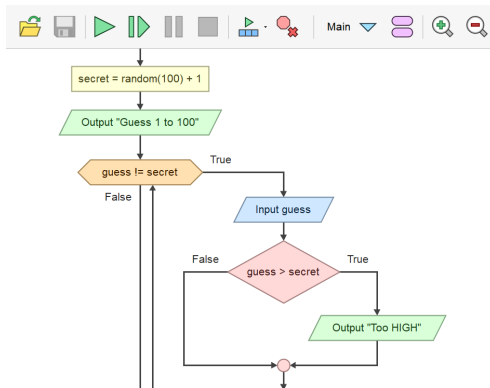
```
1 input: base
2 input: esponente
3 potenza = base
4 esegui
5   potenza = potenza*base
6   esponente = esponente - 1
7 ripeti mentre (esponente > 1)
8 output: potenza
```

Blocchi



Flowgorithm e flow.io

Flowgorithm è un programma gratuito, disponibile per Windows, per la creazione di diagrammi di flusso.



Esercizi

1. Realizza la pseudocodifica e il diagramma a blocchi di un algoritmo che riceva in input un numero e ne calcoli il fattoriale.
2. Realizza la pseudocodifica e il diagramma a blocchi di un algoritmo che riceva in input due numeri e calcoli se il primo è divisibile per il secondo.
3. Realizza la pseudocodifica e il diagramma a blocchi di un algoritmo che riceva in input due numeri e dica quale dei due è il maggiore.
4. Realizza la pseudocodifica e il diagramma a blocchi di un algoritmo che riceva in input un numero e che abbia come output la sequenza di Fibonacci con un numero di termini pari al valore ricevuto in input.

Dai codici ai linguaggi

La scrittura binaria è molto comoda e semplice da gestire per una macchina (0 = circuito chiuso, 1 = circuito aperto).

Un programma (ovvero un insieme di algoritmi) per poter essere eseguito da una macchina **deve essere scritto in linguaggio binario**.

Intuiamo tuttavia che scrivere un programma in codice binario è molto complesso per un essere umano, ed è per questo motivo che sono stati inventati i **linguaggi di programmazione**.

I linguaggi di programmazione permettono di scrivere algoritmi con un linguaggio più “vicino” a quello che parliamo.

Linguaggi di programmazione

I LDP sono particolari **linguaggi artificiali** che vengono utilizzati nella comunicazione umano-computer.

Le caratteristiche di un linguaggio di programmazione sono:

- un vocabolario ristretto (si utilizzano poche parole semplici);
- regole di costruzione delle istruzioni molto semplici e rigide;
- l'utilizzo di strutture predeterminate (come quelle viste).

Esempi di linguaggi di programmazione tra i circa 2500 esistenti: Fortran (1957), Pascal (1970), C++ (1986), Python (1991), JavaScript (1995, usato nel 98% dei siti web).

Linguaggi di basso livello

Il **linguaggio macchina** è quello direttamente compreso e utilizzato dalla CPU ed è formato solo da 0 e 1.

Un linguaggio di **basso livello** è più semplice del linguaggio macchina, ma è comunque molto lontano dai linguaggi che usiamo oggi per programmare, perché è difficile da comprendere per un umano.

Un esempio di linguaggio di basso livello è **assembly**, che usa istruzioni come:

```
05 id ADD EAX, imm3
```

Linguaggi di alto livello

I linguaggi di alto livello utilizzano un linguaggio pseudo-umano, che rende più facile la scrittura e la verifica del loro corretto funzionamento.

I linguaggi di programmazione di alto livello utilizzano come base la lingua inglese.

Esempio di codice in C++:

```
1  int num1, num2, differenza;           //variabili intere
2  cout << "Scrivi due numeri: ";      //output a schermo
3  cin >> num1 >> num2;                 //input valori delle variabili
4  differenza = num1 - num2;             //istruzione di calcolo
5  cout << "Risultato = " << differenza << endl; //output a schermo
```

Traduzione in linguaggio macchina

La macchina non può eseguire direttamente le istruzioni scritte in un linguaggio di alto livello.

È dunque necessario un “interprete” (detto **compilatore**) che traduca il **codice sorgente** (in linguaggio di alto livello) in istruzioni di macchina (cioè a basso livello).

Il codice ottenuto, che la macchina può eseguire direttamente, è detto **eseguibile** o **programma oggetto** (su Windows gli eseguibili hanno estensione **.exe**).

Cos'è un programma

Un programma è un insieme di istruzioni, codificate come **linee di codice** scritte in un certo linguaggio di programmazione.

La programmazione è la scrittura, da parte di un programmatore umano, di queste linee di codice.

L'insieme delle linee di codice costituisce il **codice sorgente** del programma.

Il codice sorgente può essere proprietario (*closed source*) oppure libero (*open source*).

Esempio di codice sorgente (Arduino)

Il codice sorgente di un programma ha una forma simile a questa:

```
1 void setup() {
2   Serial.begin(9600); //inizio trasmissione seriale per debug
3   for (int i = 0; i < notesButtRows; i++) {
4     pinMode(rowsPins[i], INPUT_PULLUP); //pullup per note
5     notesButtPState[i] = analogRead(rowsPins[i]);
6   }
7   for (int i = 0; i < encoders; i++) { //attiva pin encoders
8     pinMode(encAPins[i], INPUT_PULLUP); //output A
9     pinMode(encBPins[i], INPUT_PULLUP); //output B
10    pinMode(encButtPins[i], INPUT_PULLUP); //clic encoders
11    encPState[i] = digitalRead(encAPins[i]);
12  }
13  welcome();
14 }
```

Bug e debug

Il debugging (o debug) è l'individuazione e correzione da parte del programmatore di uno o più errori (**bug**, in italiano “baco”) rilevati nel software, direttamente in fase di programmazione oppure a seguito della fase di testing o dell'utilizzo finale del programma stesso.

I bug sono tipicamente dovuti ad **errori nella scrittura del codice sorgente** di un programma.

I bug possono essere corretti con una nuova versione del programma o attraverso una **patch**.