



INTRODUZIONE AL CORSO
fondamenti di informatica

Daniele Fadda



1. Information Designer
2. Ricercatore e tecnologo al CNR di Pisa
3. Docente di Information Visualization

Contatti

- Ricevimento: presso il CNR o online
prendere appuntamento via mail
- Email: fadda.daniele@gmail.com
mettere il tag [AlmaArtis] nell'oggetto della mail
- Sito web del corso
https://github.com/danielefadda/alma_artis_2025

A photograph of a man with long hair and a beard, wearing a grey t-shirt, working on a large black telescope mounted on a tripod in a field under a clear blue sky.

**Artisti
e coding**

Perché dovremmo programmare?

Quale linguaggio?

Dove finisce il nostro lavoro?

Posso essere un buon designer senza codice?



Dovrei diventare uno sviluppatore?

Perché dovremmo programmare?

è difficile immaginare e progettare cose interattive senza conoscere le basi della programmazione

Perché dovremmo programmare?

Mentre si gioca con un framework, un servizio o uno strumento, si aprono nuove possibilità.

Un prototipo veloce porta a una nuova idea che alla fine porta a un nuovo prototipo che porta a un'idea ancora migliore.

Code & design:

Il codice come **logica**

Il codice come uno **strumento di sketching**

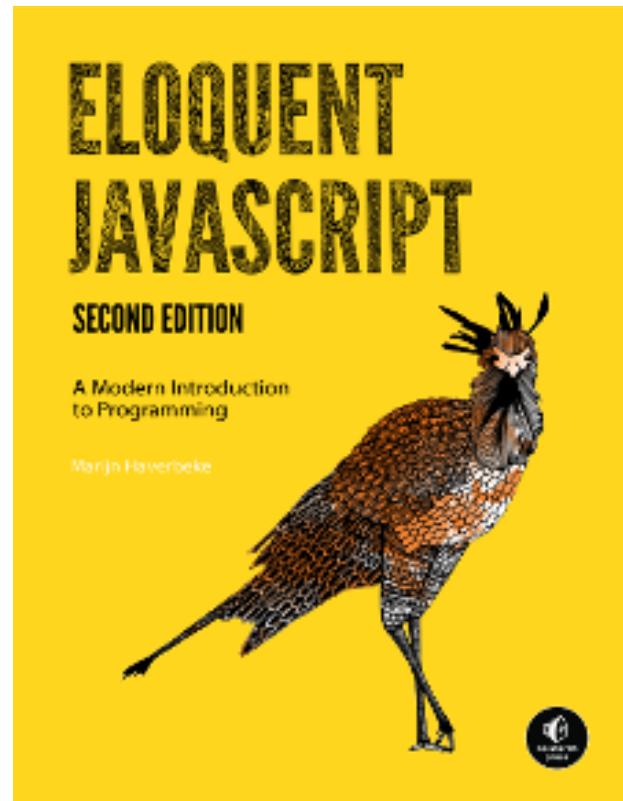
Il codice come uno **strumento generativo**

Obiettivi del Corso

- Concetti di base dell'**Informatica**
- Acquisizione di principi e strumenti di base della **programmazione**
- Elaborare la **soluzione** ad un **problema** attraverso la creazione di un **algoritmo** e la relativa codifica in un **linguaggio di programmazione**
- Introduzione alla **computer graphics**

Il linguaggio di programmazione del corso sarà **Javascript** e la libreria **p5.js**

Libri e riferimenti



Eloquent Javascript – Second Edition
Capitoli 1-4

Marijn Haverbeke
Licensed under CC license.

Available here:
<http://eloquentjavascript.net/>

Libri e riferimenti (2)



Corso MDN

MDN's mission is to provide a blueprint for a better internet and empower a new generation of developers and content creators to build it.

Mozilla Foundation

LIFE WOULD BE MUCH EASIER
IF I HAD THE SOURCE CODE

Perché questo corso?

Perché **JavaScript** è ovunque (anche dove non te lo aspetti)

Oltre il Web Development

Applicazione	Librerie/Framework	Esempi di Utilizzo
Robotica e Automazione	- Johnny-Five - Node-RED	- Controllo bracci robotici - Gestione sensori IoT
Arte Generativa	- p5.js - Three.js	- Installazioni museali interattive - Generazione di pattern frattali
Musica e Sound Design	- Tone.js - Web Audio API - Howler.js	- Sintetizzatori virtuali - Composizioni generative - Effetti audio real-time
Astronomia	- AstroJS - Celestial.js - Planet.js	- Calcolo orbite satellitari - Mappe stellari interattive - Simulazioni planetarie

Applicazione	Librerie/Framework	Esempi di Utilizzo
Archeologia Digitale	- Potree.js - OpenLayers - CesiumJS	- Modelli 3D di siti archeologici - Mappe stratigrafiche - Database reperti interattivi
Linguistica Computazionale	- Natural - Compromise - NLP.js	- Analisi semantica testi - Riconoscimento dialetti - Traduzione automatica
Neuroscienze	- BrainJS - Synaptic.js - NeuroJS	- Visualizzazione EEG - Reti neurali simulate - Interfacce neurali
Antropologia Digitale	- D3.js - Gephi.js - NetworkX.js	- Mappe migratorie - Reti sociali storiche - Analisi culturali

Dove è JavaScript

- Applicazioni Web
- Siti Web Interattivi
- Server (Node.js)
- App Mobile
- Desktop Apps
- Applicazioni IoT
- Applicazioni di Machine Learning
- Applicazioni di Realtà Virtuale e Aumentata
- Applicazioni di Intelligenza Artificiale



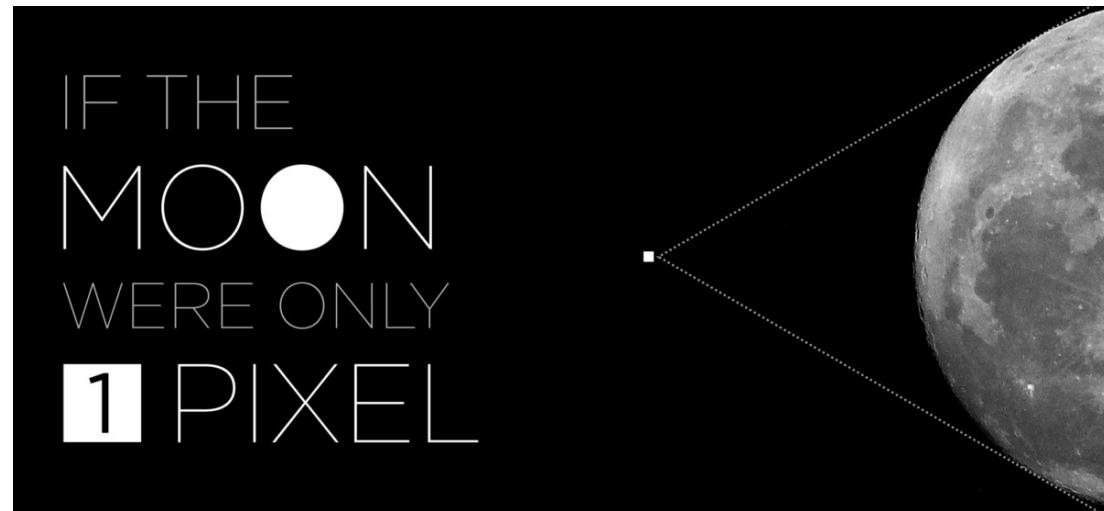


Jam Pack

Yuogo Nakamura

Webdesigner della prima epoca.
Diventa famoso con i primi lavori in Flash.
(=JavaScript)

Ora lavora per la sua agenzia di interactive design THA.jp



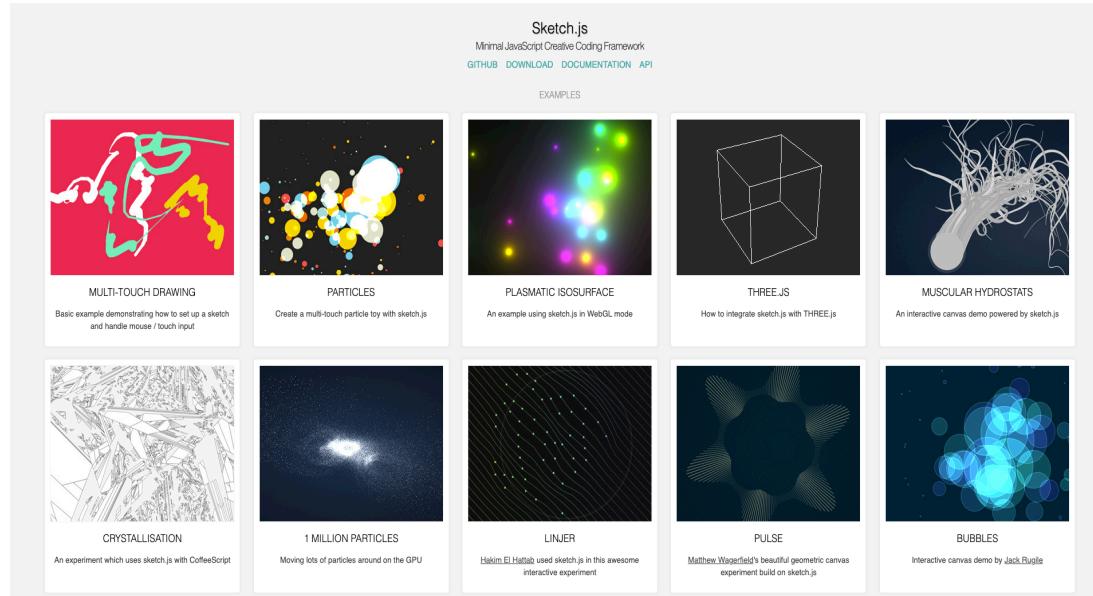
Una mappa del sistema solare che illustra la distanza tra i pianeti.

Nato come progetto personale, è stato pubblicato su centinaia di siti web, presentato nei musei, utilizzato come strumento didattico e tradotto in 16 lingue.

Il sito è stato premiato con il Webby Award nel 2016 come miglior sito scientifico.

1pixelmoon.com

Josh Worth

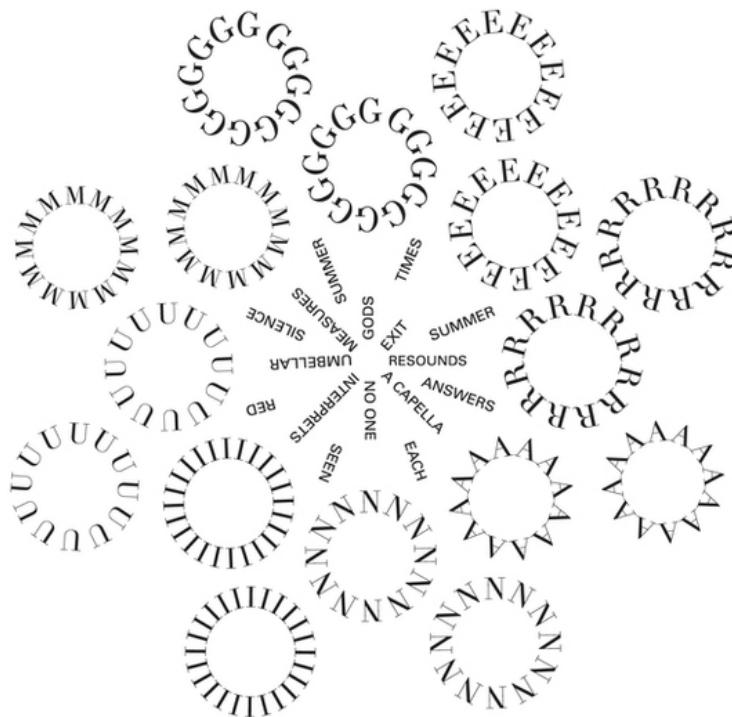


Sketch JS

Minimal JavaScript Creative Coding Framework.

Ispirato a Processing, Sketch.js è un framework JavaScript che rende semplice creare animazioni e interazioni grafiche.

Sketch JS



Geranium

di Mary Ellen Solt.

Brief: Considera la poesia concreta creata da Mary Ellen Solt e interpretala attraverso un'animazione utilizzando solo la tipografia.

Strumenti usati: After Effects, GarageBand, ActionScript (=JavaScript)



A tales of two cities

Installazione animata su tre maxischermi che seguiva migliaia di auto nei loro viaggi attraverso tre grandi città del mondo, Londra, Boston e Roma.



Informatica e programmazione

Cosa è l'informatica?

Alcune definizioni

- Scienza della **conservazione**, dell'**elaborazione** e della **rappresentazione** automatica dell'**informazione**.
- Scienza dei calcolatori e delle loro applicazioni.
- Scienza dell'**informazione**.
- Studio sistematico degli **algoritmi** che descrivono e trasformano l'informazione.

Cosa è l'informazione?

Tutto ciò che può essere rappresentato all'interno di un computer.

- Numeri, caratteri, immagini, suoni
- Comandi e sequenze di comandi che il calcolatore esegue per trasformare l'informazione

Il **computer** è lo strumento per rappresentare ed elaborare l'informazione

Cosa è un Computer?



Cacolatrice Elettronica Pisana (CEP)

Una macchina che:

- Ha un meccanismo di **input** per acquisire le richieste
- Ha un dispositivo per **memorizzare** le informazioni
- Ha la capacità di **elaborare** i dati
- Ha un dispositivo per comunicare la **risposta**

Computer vs Calcolatrice

Computer

1. Ha un meccanismo di **input** per acquisire le richieste
2. Ha un dispositivo per **memorizzare** le informazioni
3. Ha la capacità di **elaborare** i dati
4. Ha un dispositivo per comunicare la **risposta**

Calcolatrice

1. 10 tasti per le cifre da 0 a 9 per acquisire i dati
2. Una memoria interna per i risultati parziali
3. 4 tasti per le operazioni da calcolare
4. Un display per mostrare il risultato

Algoritmo: esempi

La vita quotidiana è piena di algoritmi che eseguiamo senza pensarci:

- Decidere se attraversare la strada al semaforo verde
- Montare un mobile dell'Ikea
- Prendere un caffè alla macchina a gettoni
- Sommare le bollette, sottrarlo al saldo del conto corrente, incrociare le dita...

Algoritmo: teoria

È una **sequenza** finita di passi (**istruzioni**) necessari per risolvere un problema o per raggiungere un determinato obiettivo.

Si può anche definire come la successione di operazioni da eseguire tramite il calcolatore (esecutore) per ottenere il risultato voluto.

Solitamente esistono diversi modi per risolvere un problema e quindi si possono proporre diversi algoritmi che risolvono lo stesso problema.

Esercizio 1:

Pensate a un algoritmo nella vita quotidiana

Esercizio 2:

Pensate a tutti i passaggi necessari per realizzare il vostro algoritmo



Il mio algoritmo: la frittata di cipolle

Il mio algoritmo: la frittata di cipolle

- Preparare il necessario: 2 cipolle rosse, 4 uova, sale, pepe e olio
una padella di 24 cm, un piatto fondo, un coltello affilato, una forchetta
- Tagliare la cipolla a fettine sottili
- Far scaldare l'olio nella padella
- Mettere la cipolla nella padella e far appassire
- Nel frattempo, sbattere le uova
- Mettere le uova nella padella
- Girare la frittata a metà cottura

La frittata di cipolle

- Preparare il necessario: 2 cipolle rosse, 4 uova, sale, pepe e olio
una padella di 24 cm, un piatto fondo, un coltello affilato, una forchetta
- Tagliare la cipolla a fettine sottili
- Far scaldare l'olio nella padella
- Mettere la cipolla nella padella e far appassire
- Nel frattempo, sbattere le uova
- Mettere le uova nella padella
- Girare la frittata a metà cottura

Il mio algoritmo: la frittata di cipolle

- Preparare il necessario: 2 cipolle rosse, 4 uova, sale, pepe e olio
una padella di 24 cm, un piatto fondo, un coltello affilato, una forchetta
- Tagliare la cipolla a fettine sottili
- Far scaldare l'olio nella padella
- Mettere la cipolla nella padella e far appassire
- Nel frattempo, sbattere le uova
- Mettere le uova nella padella
- Girare la frittata a metà cottura

Il mio algoritmo: la frittata di cipolle

- Preparare gli ingredienti necessari: 2 cipolle rosse, 4 uova, sale, pepe e olio
una padella di 24 cm, un piatto fondo, un coltello affilato, una forchetta
- Tagliare la cipolla a fettine sottili
- Far scaldare l'olio nella padella
- Mettere la cipolla nella padella e far appassire
- Nel frattempo, sbattere le uova
- Mettere le uova nella padella
- Girare la frittata a metà cottura

Algoritmo nel contesto informatico

A volte è complicato dividere il problema in passi elementari, che siano eseguibili da una macchina:

Esempio 1

Trovare il minimo tra i seguenti numeri

3, 12, 8, 2, 9

Esempio 2

Eseguire la somma in colonna tra due numeri

55 e 49

Esempio 1 - Algoritmo Trova Minimo

1. Inizializza minimo con il primo elemento

- **minimo = 3**
- creo una lista ordinata di numeri : **3, 12, 8, 2, 9**

2. Confronta sequenzialmente ogni elemento con il minimo

- *Confronta 3 con 12: $3 < 12$, minimo resta 3*
- *Confronta 3 con 8: $3 < 8$, minimo resta 3*
- *Confronta 3 con 2: $3 > 2$, minimo diventa 2*
- *Confronta 2 con 9: $2 < 9$, minimo resta 2*

3. Dopo aver confrontato tutti gli elementi

- Il valore minimo trovato è: **2**

Esempio 2 – Algoritmo Somma In Colonna

1. Allinea i numeri in colonna

55

49

2. Parti dalla colonna più a destra (unità)

- Prima colonna, somma: $5+9=14$
- Scrivi 4 e riporta 1

3. Passa alla colonna successiva (decine)

- Seconda colonna: $5+4+1(\text{riporto})=10$
- Scrivi 0 e riporta 1

4. Se c'è un riporto finale, scrivilo come nuova cifra più significativa

- Riporto: 1

5. Leggi il risultato da sinistra a destra:

- 104

Proprietà degli Algoritmi

- **Finito** deve terminare in un numero finito di passi
- **Eseguibile** deve essere eseguibile da una macchina con risorse limitate
- **Non ambiguo** Ogni azione deve essere interpretata in modo univoco

Se almeno una di queste proprietà non è soddisfatta, l'algoritmo non è valido.

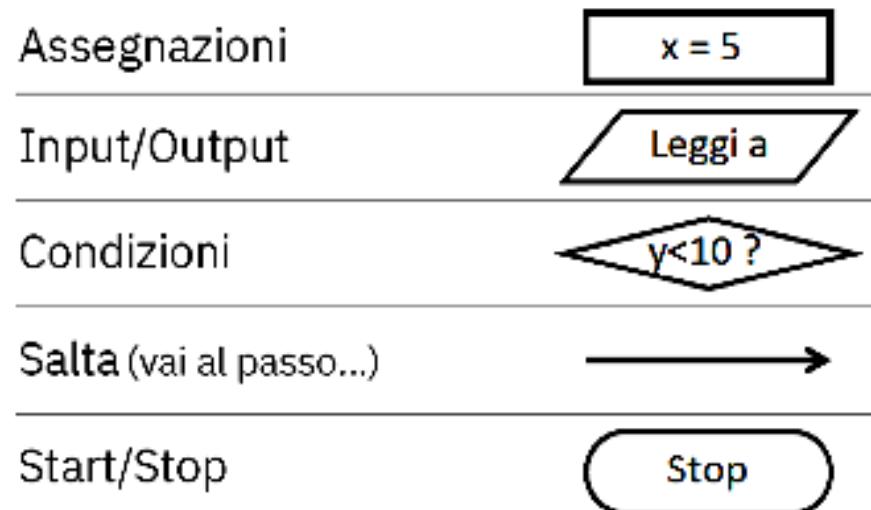
Altre proprietà di un algoritmo

- **Generalità:** l'algoritmo deve funzionare correttamente anche con piccole variazioni dell'input del problema
- **Efficienza** l'algoritmo deve essere in grado di risolvere il problema in un tempo ragionevole
- **Determinismo:** l'algoritmo deve produrre lo stesso risultato per lo stesso input

Algoritmi e programmazione

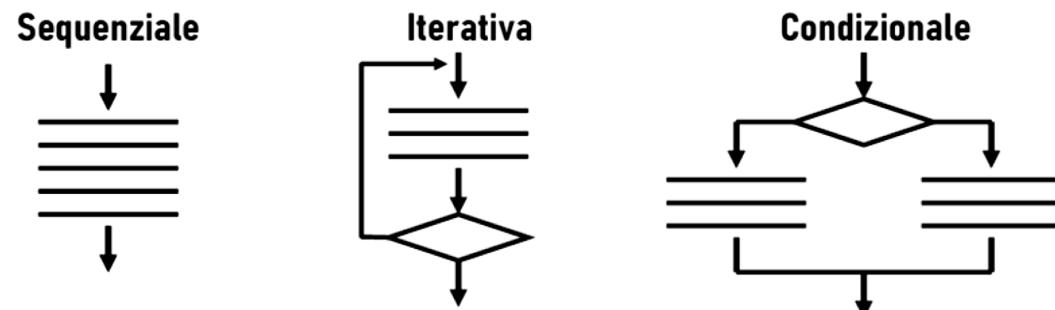
- **Algoritmo:** metodo risolutivo
- **Linguaggio di Programmazione:** linguaggio comprensibile al calcolatore
- **Programma:** sequenze di istruzioni del linguaggio che descrivono un algoritmo

Diagramma di Flusso



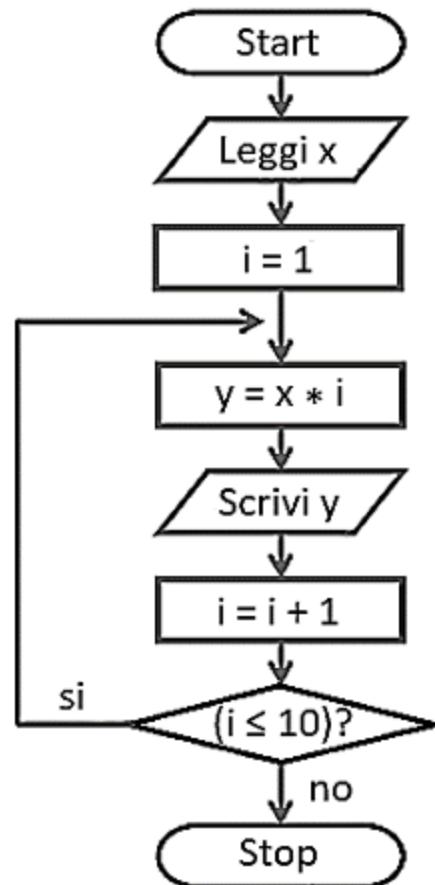
Per rappresentare in modo efficace un algoritmo sono stati sviluppati dei modelli grafici (i diagrammi di flusso) che associano alle istruzioni del programma dei simboli grafici.

Diagramma di Flusso



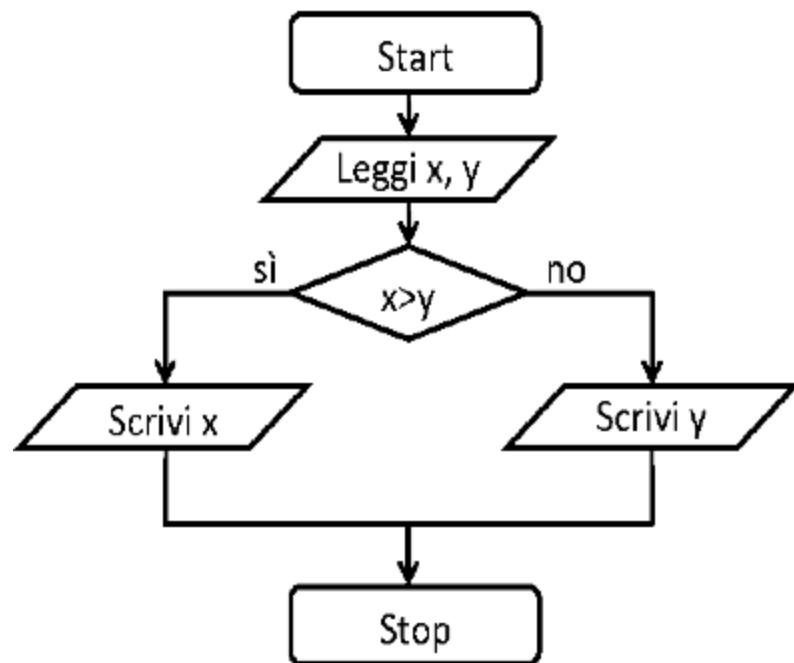
È stato dimostrato (Teorema fondamentale della programmazione strutturata di Giuseppe Jacopini e Corrado Böhm) che **ogni programma** può essere codificato attenendosi esclusivamente a **tre strutture fondamentali**: **sequenziale, iterativa e condizionale**.

Diagramma di Flusso esempio 1



Leggi x
Poni $i = 1$
Poni $y = x * i$
Scrivi y
Poni $i = i + 1$
Se $i \leq 10$ vai al passo 3 altrimenti prosegui
Fermati

Diagramma di Flusso esempio 2



Leggi x e y

x>y?

Se condizione del punto 2 è vera scrivi x

Se è falsa scrivi y

Fermati

Le fasi della programmazione

Le fasi della programmazione

1. **Specific**a: definizione del problema

- quale funzione si vuole calcolare e quali sono i dati di interesse?
- es. dati due numeri calcolarne il maggiore

2. Individuazione di un **algoritmo** (metodo risolutivo)

3. Codifica dell'algoritmo in un **linguaggio di programmazione**

4. **Esecuzione** e messa a punto (testing)

1. Specifica

- Comprendere e **definire (specificare)** il problema che si vuole risolvere
- Descrizione dello **stato iniziale** del problema (dati iniziali, input) e dello **stato finale** atteso (risultati, output)
- La specifica può essere fatta in maniera più o meno rigorosa

Esempi di specifica informale

- Es. 1: dati due numeri, trovare il maggiore
- Es. 2: dato un elenco telefonico e un nome, trovare il numero di telefono corrispondente
- Es. 3: data la struttura di una rete stradale e le informazioni sui flussi dei veicoli, determinare il percorso più rapido tra due posizioni A e B

Note

La descrizione è ambigua o imprecisa (es. 2)

La descrizione non fornisce un metodo risolutivo (es. 3)

2. Algoritmo

Dopo la specifica occorre individuare un algoritmo o **metodo risolutivo** che permetta di ottenere i risultati attesi.

In generale, in questa fase occorre:

- Individuare una soluzione
- Dimostrare che la soluzione è corretta
- Tra più soluzioni, scegliere quella ottimale

3. Codifica

- Rappresentare l'algoritmo e l'informazione di interesse in un linguaggio di programmazione
- Il risultato è un programma eseguibile dal calcolatore
- Occorre rappresentare
 - l'algoritmo ⇒ programma
 - le informazioni iniziali ⇒ dati in ingresso (input)
 - le informazioni finali ⇒ dati in uscita (output)
 - le informazioni usate dall'algoritmo ⇒ dati ausiliari

4. Esecuzione

Due sono le modalità principali di esecuzione dei programmi

Compilazione

"libro tradotto da una lingua all'altra"

Programma tradotto in linguaggio
macchina

Esecuzione diretta dal calcolatore

Es: C, C++, Java (parzialmente)

Interpretazione

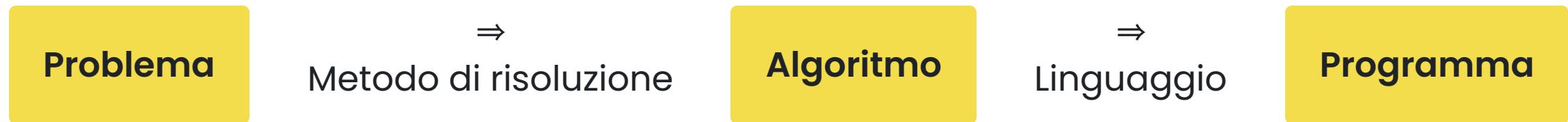
"traduttore simultaneo"

Programma eseguito da un interprete
Traduzione ed esecuzione simultanea

Es: Python, JavaScript

Linguaggi di programmazione e JavaScript

Algoritmo e Linguaggio di Programmazione



Problema: prodotto di n numeri

1. Specifica del problema:

- Input: N numeri $\{x_1, x_2, \dots, x_N\}$
- Output: un numero P tale che $P = x_1 * x_2 * \dots * x_N$

2. Specifica dell'algoritmo

$P = 1$

$i = 1$

finché ($i \leq N$)

$P = P * x_i$

$i = i + 1$

I linguaggi di programmazione sono tutti equivalenti

Linguaggio macchina:

```
00110101 00000000 00000000 00000000  
00111111 00000000 00000000 00001010  
01110100 00000000 00000000 00001111
```

Linguaggi di Alto Livello:

```
let total = 0;  
let count = 1;  
while (count <= 10) {  
    total += count;  
    count += 1;  
}
```

N.B. Il risultato è lo stesso

Linguaggio macchina

Il linguaggio **direttamente eseguibile da un calcolatore** si chiama linguaggio macchina

È un linguaggio **poco comprensibile per gli umani**

Le operazioni disponibili sono molto semplici

Sono **specificate in notazione binaria**: sequenze di 1 e 0

Lo codifica di una **funzione** complessa richiede la scrittura di un **lungo programma**, spesso incomprensibile

In caso di errori o malfunzionamenti, è difficile trovare gli eventuali errori (**debugging**)

Linguaggi di Alto Livello

Sono linguaggi che permettono una **leggibilità** e una comprensione **più immediata** degli algoritmi che codificano

Sono compatti, comprensibili, modificabile e **mantenibili**

I programmi di un linguaggio di alto livello **devono essere tradotti in linguaggio macchina** per essere eseguiti dal calcolatore

somma dei primi 10 numeri

```
00110001 00000000 00000000  
00110001 00000001 00000001  
00110011 00000001 00000010  
01010001 00001011 00000010  
00100010 00000010 00001000  
01000011 00000001 00000000  
01000001 00000001 00000001  
00010000 00000010 00000000  
01100010 00000000 00000000
```

somma dei primi 10 numeri: assembly

```
; [0] -> total  
; [1] -> count  
; [2] -> temp per confronto
```

```
MOV [0], 0    ; 1. Inizializza total a 0  
MOV [1], 1    ; 2. Inizializza count a 1
```

...

...

loop:

```
    MOV [2], [1]    ; 3. Copia count in temp  
    SUB [2], 11     ; 4. Sottrai 11 da temp  
    JZ end         ; 5. Se temp = 0, vai a end  
    ADD [0], [1]    ; 6. total += count  
    INC [1]         ; 7. count++  
    JMP loop       ; 8. Torna a loop
```

end:

```
    OUT [0]        ; 9. Output total
```

somma dei primi 10 numeri: Porth

Set “total” to 0.

Set “count” to 1.

[loop]

Set “compare” to “count”.

Subtract 11 from “compare”.

If “compare” is zero, continue at [end].

Add “count” to “total”.

Add 1 to “count”.

Continue at [loop].

[end]

Output “total”.

somma dei primi 10 numeri: JavaScript

```
var total = 0, count = 1;  
while (count <= 10) {  
    total += count;  
    count += 1;  
}  
console.log(total);  
// → 55
```

somma dei primi 10 numeri: JavaScript semplificato

```
console.log(sum(range(1, 10)));
// → 55
```

Perché JavaScript?

- Linguaggio di programmazione **versatile**
- **Facile da imparare**
- Molto utilizzato perché presente **in tutti i browser**
- Adatto per lo sviluppo **web (ma non solo)**
- Ha una grande **comunità** alle spalle
- Ha **molte risorse** che sono disponibili gratuitamente

JavaScript: Una Panoramica

JavaScript (JS) è un linguaggio di programmazione interpretato (linguaggio di scripting) **Creato nel 1995** e da Brandon Eich (USA) **in 10 giorni**, prendendo in prestito molte delle migliori caratteristiche da vari altri linguaggi di programmazione.

Obiettivo: dare ai web designer la possibilità di **interagire con i diversi elementi della pagina** (immagini, form, link, ecc.).

JavaScript: Una Panoramica

Il nome originale di JS era **Mocha**. È stato rinominato **LiveScript** con la prima versione beta del browser Netscape Navigator ed è stato poi cambiato in **JavaScript** quando è stato implementato nel browser Netscape 2 nel 1995.

Microsoft ha introdotto un clone esatto di JavaScript in Internet Explorer, chiamandolo **Jscript** al fine di aggirare i problemi di marchio.

JavaScript **non è** Java

Java e JavaScript sono due linguaggi di programmazione diversi tra loro, seppure abbiano alcune caratteristiche in comune. Per esempio una notevole differenza è che **Java è compilato**, mentre **JavaScript è interpretato**. L'interprete di JS è incorporato nei web browser

Javascript è un linguaggio di programmazione Object-based. Esso è stato ideato per rendere dinamiche ed interattive le pagine web.

Esempio Pratico

```
// Calcolo della somma dei numeri da 1 a 10
let total = 0;
let count = 1;

while (count <= 10) {
    total += count;
    count += 1;
}

console.log(total); // Stampa: 55
```

Best Practices

- Scrivere codice leggibile
- Commentare il codice
- Mantenere la semplicità
- Testare regolarmente
- Imparare dagli errori

Come Iniziare

1. Scegliere un editor di testo
2. Aprire la console del browser
3. Provare esempi semplici
4. Esercitarsi regolarmente
5. Risolvere esercizi pratici e variarli
6. Costruire piccoli progetti

FINDING OUT YOUR CHILD IS A PROGRAMMER



Prova ora!

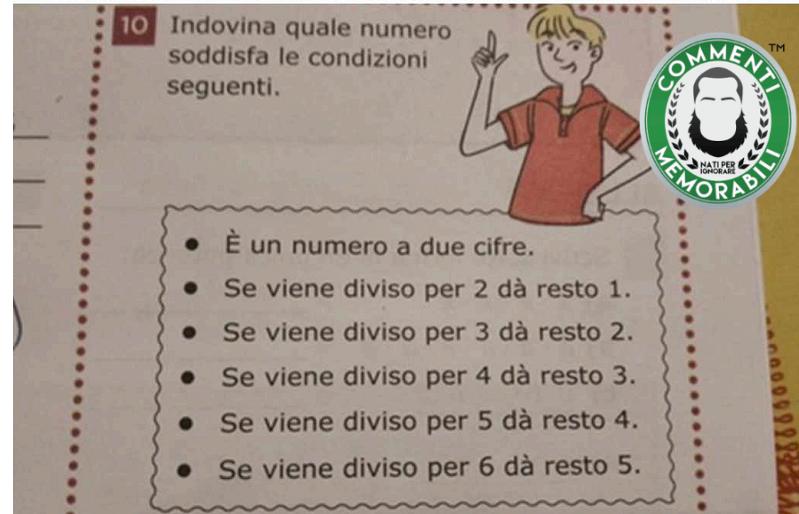
1. Apri il **browser**
2. Apri la **console** (F12)
3. Scrivi il seguente **codice**:

```
console.log("Hello, World!");
```

Domande?

Bonus slide: un esempio

Problema delle elementari che ti farà sentire scemo/a.



Salvatore Caronia
La maggior parte delle persone in questo momento
582 😅👍




Jael Donadello
La matematica è ora che cresca e si risolva i problemi da sola come tutti
335 😅👍



```
n = 10;  
while (n < 100) {  
    if (  
        n % 2 === 1 &&  
        n % 3 === 2 &&  
        n % 4 === 3 &&  
        n % 5 === 4 &&  
        n % 6 === 5  
    )  
    {  
        console.log("il numero cercato è " + n);  
        break;  
    } else {  
        console.log(  
            "il numero cercato non è " + n  
            + " proseguo con il controllo"  
        );  
    }  
    n++;  
}
```