Logiche di programmazione e basi di programmazione a oggetti, Matteo-Martinelli TECH-UP ACCELERATOR

Simone Capodivento

January 2026

Contents

Se vuoi diventare il nuovo MacGyver:

Figure 1: Lucas Till AKA MacGyver

questo corso fa per te :-)

1 Introduzione

Logiche di programmazione e basi di programmazione a oggetti.

2 Il mondo pieno di problemi

$\boxed{L'informatica}$ serve a risolvere problemi, l'uomo deve alimentarsi, questo è un esempio di problema possibile.

3 Problem solving

Questi sono gli step per risolvere un problema like a pro:

- Identificarlo
- Cosa ha causato tale problema e come si risolve
- Pensare a possibili soluzioni (*Prototyping*)
- Valutare Pros e Cons
- Prendere azione
- Valutare il risultato

4 Approccio computer-based

Abbiamo sotto mano uno dei mezzi più potenti con il quale interagiremo nella nostra vita, ma non risolve tutti i problemi. Lui ha capacità sovraumane di calcolo e quindi il digitale implica automazione.

5 Tipi di problemi

$\boxed{L'informatica}$ serve per gestire le informazioni. Un problema ben definito ha chiaro:

- Quali sono gli input
- Quale l'output desiderato
- Quali sono i vincoli

Un problema con logica compatibile deve avere una sequenza di passaggi logici, un algoritmo che porti dalla situazione iniziale alla soluzione; un esempio è l'algoritmo di Dijkstra.

6 Dati elaborati

Il formato dei dati deve essere leggibile dai computer. Quindi deve essere:

- Strutturato
- Accessibile
- Sufficente

Il computer fa ciò:

- Elabora dati rapidamente
- Gestire grandi volumi di dati
- Lavora 24h/24h
- Mantenere la precisione nei calcoli ripetitivi
- Scalabilità efficace (risolvere versioni più grandi dello stesso problema)

[1]

[1] "Pareto superiore" (o miglioramento paretiano) si riferisce a una situazione in cui si può migliorare la condizione di almeno una persona senza peggiorare quella di nessun'altra.

7 Approccio euristico

Qui elencati ci sono problemi di elaborazione dei dati:

- Calcola la media dei voti a scuola
- Cerca una parola specifica in un milione di documenti
- Convertire un'unità di misura in un'altra

8 Problemi di ottimizzazione

Alcuni problemi per esempio sono:

- Trovare il percorso più veloce per una consegna
- Assegnare in modo efficente le risorse limitate
- Pianificare un programma scolastico che soddisfi il maggior numero di vincoli

$\boxed{L'informatica}$ *inizia con questa consapevolezza fondamentale: L'informatica è uno strumento straordinariamente forte, ma non è una risposta universale.*

9 Flow

$\boxed{L'informatica}$ è un'attività sociale. Pareto 80/20. L'80 sta per il tempo per modellare il problema e la soluzione. Il 20 sta per il tempo per implementarlo.

- Analisi del problema
- Sviluppo dell'algoritmo
- Traduzione dell'algoritmo in istruzioni comprensibili al computer, quindi i linguaggi di programmazione.

La scrittura di codice consuma molto tempo ed è costosa.

Il flow state è quel momento più o meno lungo nel quale sei completamente immerso in un'attività e ti dimentichi che sei un essere umano quindi non mangi o vai alla toilette.

10 Linguaggi di programmazione

Il primo linguaggio di programmazione che vedremo è quello binario composto solamente da 0 1 0 1 1 0 ecc. Ormai è obsoleto, la lingua italiana è troppo ambigua per poter essere compresa da un computer.

11 C

Fu il linguaggio più impattante ed ha queste caratteristiche:

- Procedurale
- Low-level, ottimizzato per memory management e performance operations
- Compilato
- Portabile
- Minimalista
- Permette il controllo diretto sulla memoria e hardware, è ideale per il system programming, operating system development, drivers ed embedded software

I driver sono pezzi di codice che interagiscono con pezzi analogici. Con embedded software intendiamo hardware systems che fanno un solo task, come esempio ho scelto Arduino che contiene codice in C++.

12 C++

C++ è un linguaggio multi-paradigma, è sia procedurale che object-oriented. Ha anche queste caratteristiche:

- Può essere considerato un'estensione di C con l'aggiunta del supporto all'OOP
- Compilato
- Come C offre la manipolazione della memoria e high-performance programming
- Comunemente usato per i videogiochi, applicazioni desktop, graphic rendering engines e systems requiring high-performance

13 C#

Lui è completamente OO. Sviluppato da Microsoft, fa parte del .NET ecosystem. Queste sono le sue caratteristiche:

- Utilizza garbage collector per la gestione automatica della memoria.
- Compilato
- Possiede una sintassi semplice simile a C++
- Supporta features avanzate come async/await per la asynchronous programming.

14 Java

Write once, run everywhere

Famoso per essere fondante per molto altro che vedremo poi. Possiede queste caratteristiche: Questo linguaggio di programmazione è OO e cross-platform programming language.

- Possiede il garbage collector
- Ha una sintassi simile a C
- Possiede una library standard per il networking, I/O graphical interface.
- Tipicamente viene usato per applicazioni enterprise-level, sviluppo Android, web server e distributed application

15 Python

Esso possiede la Dynamically-typed features, faccia da usare, multi-paradigm, procedurale, OO funzionale. Come caratteristiche possiede:

- Sintassi chiara, ideale per il prototyping e lo sviluppo veloce
- interpretato
- Possiede garbage collector
- Ha una vista libreria standard, possiede molte libraries di terze parti per la data analysis, machine learning e web development

16 JavaScript

Questo possiede dynamic, multi-paradigm (procedurale, OO, funzionale)come caratteristiche e:

- Nato per esugire codice *client-side* nel browser, è oggi utilizzato anche *server-side* (vedi Node.js)
- interpretato, alcune volte JIT-compiled (per migliorare le performance)
- Dinamically typed, non è necessario dichiarare esplicitamente il tipo di variabile, ciò potrebbe cambiare il *runtime*
- Permette l'asynchronous programming
- Tipicamente viene usato per lo sviluppo web, front-end, back-end e per gli environments full-stack

17 GO

Anche chiamato *Golang*, è un linguaggio compilato di proprieta di GoOgLe

- Possiede una sintassi semplice e chiara simile al C
- Garbage collection
- molto veloce nella compilazione
- Built-in concerruncy per poter fare operatzioni parallele
- Ha una libreria standard ampia, alternativa moderna per i sistemi back-end
- Usato per applicazioni server, servizi di rete e cloud computing[2]

18 Ruby

Un linguaggio OO e dinamico che possiede queste caratteristiche:

- Ha una sointassi facilmente leggibile e chiara
- Possiede garbage collection
- Viene usato principalmente per lo sviluppo we, *automation scripts* e prototyping

19 Programmi che traducono programmi

In questa sezione si approfondire il concetto di compilatore e le sue differenze rispetto agli interpreti:

19.1 Compilatori and Interpreti

I *compilatori* traducono interamente un programma, facendo risulatre come output un' *Object program* o un' *executable*. Gli *interpreti* d'altro canto traducono solamente un'istruzione alla volta in un liguaggio comprensibile alla macchina e lo esegue. Il processso di traduzione avviene ogni volta che si preme *Run*.

20 Concetti e strumenti utili

Nell'Informatica tutto si basa su questo: Input→execution→output Dove l'input sono i dati che noi diamo in pasto:

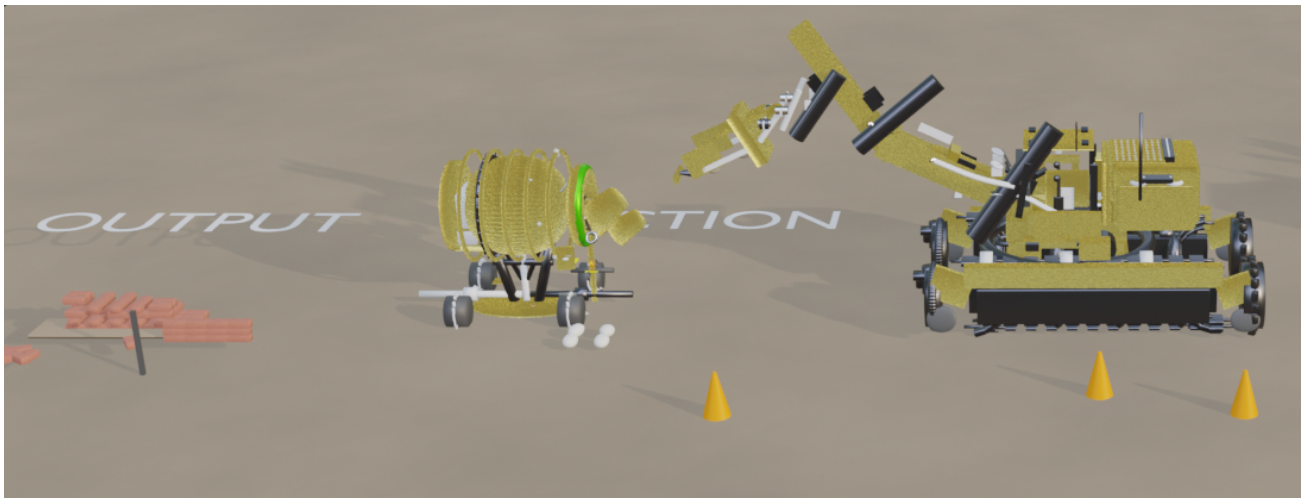

Figure 2: *Input → execution → output*

Ho fatto una rappresentazione 3D in blender di un mezzo che rappresentere l'input che noi diamo ovvere il materiale scavato da esso. Successivamente Il materiale viene inserito nell'esecutore che nel nostro caso è un Cement Mixer che ha il

[2]Ambiguità: nella programmazione riguarda la grammatica scorretta e alla disinformazione che può portare

compito di eseguire la miscela dei materiali, Infine abbiamo l'Output Ovvero un muro di mattini costruito grazie al lavoro eseguito in precedenza.Questi esempi sono Digital artifac[3]

21 Carte CRC Class-Resonsibility-Collaborator (Connetion) Cards

Essenzialmente sono semplici carte per la designazione OO systems, utili specialmente per la phase preliminare di analisi e design. Loro ci aiutano a chiarire il compito di ogni classe e con quale altra classe dovrebbe collaborare, anche prima di scrivere i diagrammi UML o di scrivere il codice.

22 Bibliografia

.

References

[1] Martinelli, M. (2026). *IFTS 2026 - Software Development with AI Tools*. Repository GitHub. Disponibile all'indirizzo: https://github.com/ifts-2026-sw-dev-with-ai-tools [Consultato il: 10 gennaio 2026]

[3]Un' Digital Artifcat è un oggetto digitale, un'oggetto è una struttura con dei limiti, attributi e metodi