

Corso Web MVC

Introduzione

Emanuele Galli

www.linkedin.com/in/egalli/

Informatica

- Informatique: information automatique
 - Trattamento automatico dell'informazione
- Computer Science
 - Studio dei computer e come usarli per risolvere problemi in maniera corretta ed efficiente

Computer

- Processa informazioni
- Accetta input
- Genera output
- Programmabile
- Non è limitato a uno specifico tipo di problemi

Hardware, Software, Firmware


- Hardware
 - Componenti elettroniche usate nel computer
 - Disco fisso, mouse, ...
- Software
 - Programma
 - Algoritmo scritto usando un linguaggio di programmazione
 - Codice utilizzabile dall'hardware
 - Processo
 - Programma in esecuzione
 - Word processor, editor, browser, ...
- Firmware
 - Programma integrato in componenti elettroniche del computer (ROM, EEPROM)
 - UEFI / BIOS: avvio del computer
 - Avvio e interfaccia tra componenti e computer

Sistema Operativo

fa parte del software

- Insieme di programmi di base
 - Rende disponibile le risorse del computer
 - All'utente finale mediante interfacce
 - CLI (Command Line Interface) / GUI (Graphic User Interface)
 - Agli applicativi
 - Facilità d'uso vs efficienza
- Gestione delle risorse:
 - Sono presentate per mezzo di astrazioni
 - File System ---> FILE: COLLEZIONE DI 0 E 1
 - Ne controlla e coordina l'uso da parte dei programmi
- Semplifica la gestione del computer, lo sviluppo e l'uso dei programmi

Problem solving

- Definire chiaramente le **specifiche** del problema
 - Es: calcolo della radice quadrata. Input? Output?
 - Vanno eliminate le possibili ambiguità
- Trovare un **algoritmo** che lo risolva 
- Implementare correttamente la soluzione con un linguaggio di programmazione
- Eseguire il programma con l'input corretto, in modo da ottenere l'output corretto

Algoritmo

- Sequenza di istruzioni che garantisce di dare il risultato di un certo problema
 - Ordinata, esecuzione sequenziale (con ripetizioni)
 - Operazioni ben definite ed effettivamente eseguibili
 - Completabile in tempo finito
- Definito in linguaggio umano ma artificiale
 - Non può contenere ambiguità
 - Deve essere traducibile in un linguaggio comprensibile dalla macchina

Le basi dell'informatica

- Matematica

- L'algebra di George Boole ~1850

- Notazione binaria



- La macchina di Alan Turing ~1930

- Risposta all'Entscheidungsproblem (problema della decisione) posto da David Hilbert
 - Linguaggi di programmazione Turing-completi

- Ingegneria

- La macchina di John von Neumann ~1940

- Descrizione dell'architettura tuttora usata nei computer: Input, Output, Memoria, CPU

Algebra Booleana

- Due valori
 - false (0)
 - true (1)
- Tre operazioni fondamentali
 - AND (congiunzione)
 - OR (disgiunzione inclusiva)
 - NOT (negazione)

A	B	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

A	NOT
0	1
1	0

Linguaggi di programmazione

- Linguaggio macchina
 - È il linguaggio naturale di un dato computer
 - Ogni hardware può averne uno suo specifico
 - Istruzioni e dati sono espressi con sequenze di 0 e 1
 - Estremamente difficili per l'uso umano
- Linguaggi Assembly
 - Si usano abbreviazioni in inglese per le istruzioni
 - Più comprensibile agli umani, incomprensibile alle macchine
 - Appositi programmi (assembler) li convertono in linguaggio macchina

BIT: la base di partenza è sempre 2 (PERCHE' I VALORI SONO SEMPRE 1 E 0 come nel sistema binario) quindi nel momento in cui abbiamo 4 valori, questi li otteniamo tramite 2 elevato 2 dove otteniamo 4. Allo stesso modo avremo 2 elevato 3, 2 elevato 4 ecc ecc. 2 VALORI CON 3 COMBINAZIONI DIVERSE. Ad oggi siamo arrivati ad avere una memoria di 64 bit quindi avremo 2 alla 64.

per lavorare su numeri interi negativi prendiamo il bit più a sinistra e inseriamo il numero 1 per indicare il negativo. quindi invece che 64 avremo 63 e così via.

UTF: tabella che servono per convertire i caratteri utili. Possiamo avere UTF-8 oppure UTF-16 ecc in base a quanti bit abbiamo a disposizione.

Variabile

- Locazione di memoria associata a un nome, contiene un valore
- Costante: non può essere modificata dopo la sua inizializzazione
- Una singola locazione di memoria può essere associata a diverse variabili (alias)
- Supporto a tipi di variabili da linguaggi di:
 - “basso livello” → legati all’architettura della macchina
 - “alto livello” → tipi complessi
 - script → runtime

I DATI NON SONO ASSOCIATI IN NESSUN MODO MA VENGONO ELABORATI AL MOMENTO. NON ABBIAMO UN CONTROLLO SULLA VARIABILE.

CREO UNA VARIABILE UNICA (ESEMPIO, TEMPERATURES) E DICO CHE E' UNA LISTA IN CUI INSERISCO I MIEI VALORI. I VALORI LI INSERISCO ALL'INTERNO DI PARENTESI QUADRE []. COME VENGONO MEMORIZZATI? PRENDOP UN BLOCCO DI MEMORIA ADEGUATO CON UN TOTALE DI CELLE, OGNUNA DA 64 BIT.

PER TROVARE UN DETERMINATO VALORE UTILIZZO DEGL IINDICI PER INDICARE ALLA MACCHINA CHE DEVE ANDARE A TROVARE SOLO QUEL DETERMINATO VALORE.

Array = COLLEZIONE DI VARIABILI

- Struttura dati comune a molti linguaggi di programmazione
- Basata sul concetto matematico di vettore, nel senso di matrice monodimensionale
- Collezione di elementi (dello stesso tipo) identificati da un indice
 - Il primo elemento ha indice 0 in alcuni linguaggi, 1 in altri (e anche n in altri ancora)
- Gli elementi sono allocati in un blocco contiguo di memoria, il che permette accesso immediato via indice ai suoi elementi

Linguaggi di alto livello

- Molto più comprensibili degli assembly
- Termini inglesi e notazioni matematiche
- Possono essere espressi in forma
 - **imperativa**: si indica cosa deve fare la macchina
 - **dichiarativa**: si indica quale risultato si vuole ottenere COME HTML (NON E' UN LINGUAGGIO DI PROGRAMMAZIONE)
- A seconda di come avviene l'esecuzione si parla di linguaggi
 - **compilati**: conversione del codice in linguaggio macchina, ottenendo un programma eseguibile COMPILER (ASSOCIATO AL LINKER): MACCHINARIO CHE SERVE ALLA MACCHINA ALTRIMENTI NON RIESCE A CAPIRE IL CODICE COMPENSIBILE SOLO AGLI UMANI. CONVERTENDOLO VIENE FUORI UNA SERIE DI 0 E 1 E LA COMPILER A' SOLO PER UNA SPECIFICA MACCHINA COME AD ESEMPIO WINDOWS O IOS.
 - **interpretati**: il codice viene eseguito da appositi programmi IL CODICE VIENE DATO ALL'UTENTE, CHE LO MANDA AD UN INTERPRETE E LI VIENE ESEGUITO. IL BROWSER O QUALSIASI ALTRO INTERPRETE LEGGE IL CODICE (COME PER JAVA SCRIPT), LO ESEGUE E LO INTERPRETA.

Istruzioni

- Operazioni **sequenziali**
 - Chiedono al computer di eseguire un compito ben definito, poi si passa all'operazione successiva
- Operazioni **condizionali**
 - Si valuta una condizione, il risultato determina quale operazione seguente verrà eseguita
- Operazioni **iterative**
 - Richiede di ripetere un blocco di operazioni finché non si verifica una certa condizione – se ciò non accade: loop infinito

Flow chart vs Pseudo codice

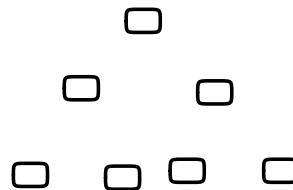
- Diagrammi a blocchi – flow chart
 - L'algoritmo viene rappresentato con un grafo orientato dove i nodi sono le istruzioni
 - Inizio e fine con ellissi
 - Rettangoli per le operazioni sequenziali (o blocchi)
 - Esagoni o rombi per condizioni
- Pseudo codice
 - L'algoritmo viene descritto usando l'approssimazione un linguaggio ad alto livello, si trascurano i dettagli, ci si focalizza sulla logica da implementare

Complessità degli algoritmi

- “O grande”, limite superiore della funzione asintotica
 - Costante $O(1)$
 - Logaritmica $O(\log n)$
 - Lineare $O(n)$
 - Linearitmico $O(n \log n)$
 - Quadratica $O(n^2)$ – Polinomiale $O(n^c)$
 - Esponenziale $O(c^n)$
 - Fattoriale $O(n!)$
- Tempo e spazio
- Caso migliore, peggiore, medio

ALBERI BST [BINARY SORTED TREE] : ALBERI CHE SERVONO PER TENERE IN ORDINE LE INFORMAZIONI. PER CERCARE ALL'INTERNO DI QUESTI ALBERI E' SEMPLICE ATTRAVERSO UNA ESPRESSIONE ALGORITMICA TRAMITE UNA FUNZIONE ESPONENZIALE.


OGNI LIVELLO DELL'ALBERO CRESCE IN MODO ESPONENZIALE E IN OGNI LIVELLO CI SONO SEMPRE MULTIPLI DI 2.



Algoritmi di ordinamento

- Applicazione di una relazione d'ordine a una lista di dati
 - Naturale → crescente (alfabetico, numerico)
- Utile per migliorare
 - l'efficienza di altri algoritmi
 - La leggibilità (per gli umani) dei dati
- Complessità temporale
 - $O(n^2)$: algoritmi naive
 - $O(n \log n)$: dimostrato ottimale per algoritmi basati su confronto
 - $O(n)$: casi o uso di tecniche particolari

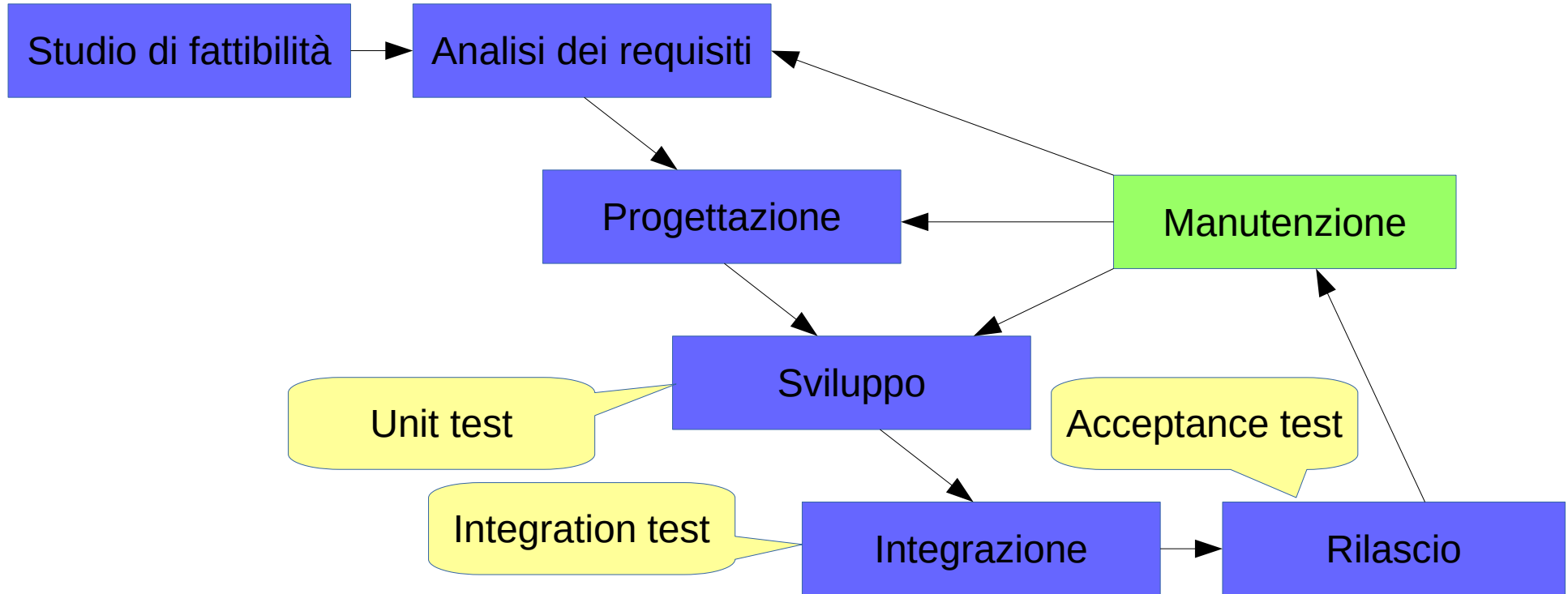
Ingegneria del software

- Approccio sistematico alla creazione del software
 - Struttura, documentazione, milestones, comunicazione e interazione tra partecipanti
- Analisi dei requisiti
 - Formalizzazione dell'idea di partenza, analisi costi e usabilità del prodotto atteso
- Progettazione
 - Struttura complessiva del codice, definizione architetturale
 - Progetto di dettaglio, più vicino alla codifica ma usando pseudo codice o flow chart
- Sviluppo
 - Scrittura effettiva del codice, e verifica del suo funzionamento via **unit test** 
- Manutenzione
 - Modifica dei requisiti esistenti, bug fixing

Unit Test

- Verificano la correttezza di una singola “unità” di codice
 - Mostrano che i requisiti sono rispettati
- Verifica
 - Casi base (positivi e negativi)
 - Casi limite
- Ci si aspetta che siano
 - Ripetibili: non ci devono essere variazioni nei risultati
 - Semplici: facile comprensione ed esecuzione
 - E che offrano una elevata copertura del codice

Modello a cascata (waterfall)



Modello agile



Software Developer

- Front End Developer
 - Pagine web, interazione con l'utente
 - HTML, CSS, JavaScript
 - User Experience (UX)
- Back End Developer
 - Logica applicativa
 - Java, C/C++, Python, JavaScript, SQL, ...
 - JavaEE, Spring, Node, DBMS, ...
- Full Stack Developer
 - Sintesi delle due figure precedenti