

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

FOLDER : contiene altri folder e file
FILE : unità minima di memorizzazione della memoria di massa
possiede un nome detto "file name"

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

FOLDER : contiene altri folder e file
FILE : unità minima di memorizzazione della memoria di massa.
possiede un nome detto "file name"
il file system si identifica con C : / se voglio accedere ai file parto dalla
ruta (ossia C : /) in cui trovo varie cartelle con all'interno altre sottocartelle
formando un albero.

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

una volta trovato l'algoritmo, devo trasformare l'algoritmo in un programma che contiene un codice scritto in linguaggio macchina così che poi essa lo possa tradurre ed eseguire così che l'utente ottenga il suo risultato.

Algoritmo

- Sequenza di istruzioni che garantisce di dare il risultato di un certo problema
 - è una sorta di ricetta ordinata da seguire alla lettera per poter arrivare alla fine dove si ottiene il risultato. L'esecuzione è sequenziale e a volte ci possono essere dei loop dove si ripetono determinate operazioni. Le istruzioni sono le parti minime che servono per poter poi creare l'algoritmo.
 - 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
 - La macchina deve capire cosa noi le stiamo dicendo ed è per questo che il linguaggio rigorizzato deve essere artificiale, ossia un linguaggio che può scrivere un programmatore informatico.
 - 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



LINGUAGGI TURING-COMPLETI:

- Gestisce un input che elabora e da un output
- per essere un linguaggio di programmazione turing-completo ci devono essere dei blocchi di istruzioni
- Devono esserci delle variabili [le variabili sono in memoria e quindi il programma può maneggiare con esse]
- La macchina deve prendere delle decisioni ossia ha bisogno di una istruzione condizionale come if e else
- Poter eseguire i loop attraverso il comando for/while cioè eseguire più volte le stesse istruzioni

- Ingegneria

- La macchina di John von Neumann ~1940

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

Le istruzioni vengono date alla CPU. Una volta create le informazioni vengono passate al bus di sistema che le trasporterà alla CPU. Nella memoria inseriamo le informazioni.

Algebra Booleana

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

ESEMPIO :

- scegliere tutti i maglioni che hanno i pois e le righe [AND]
- scegliere tutti i maglioni che hanno i pois o le righe [OR]
- scegliere tutti i maglioni che non hanno le righe [NOT]

IMPORTANTI DA SAPERE

| 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

SCOPO DELL'ALGORITMO : avere una ricetta rigorosa da farlo capire al computer per poi poterlo tradurre in linguaggio macchina.

- 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

Permettono di tradurre con un linguaggio umano un linguaggio macchina che altrimenti sarebbe incomprensibile da capire, essendo una sequenza esclusiva di 0 e 1. Il linguaggio assembly possiamo definirlo il nostro traduttore. L'assembly insieme all'assembler si occupano di tradurre il linguaggio e spedirlo alla macchina che eseguirà poi.

- 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.

= UNA VARIABILE CHE CONTIENE UNA SERIE
DI VALORI

ex : creo una variabile unica con n celle in cui inserisco tutti i valori delle temperature minime di un determinato anno. In questo caso avremo una variabile unica con 365 celle in cui inserire i valori.

Array

- 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

STRINGA : è una sorta di array di caratteri quindi una sequenza di caratteri che ci permette di parlare con un linguaggio macchina

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 per la precisione noi indichiamo e diamo un ordine alla CPU che lo eseguirà
 - **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. IL CODICE SORGENTE E' IL CODICE CHE SCRIVIAMO NOI, IL COMPILATORE PRENDE IL CODICE SORGENTE E LO TRASFORMA IN CODICE MACCHINA CHE E' ESEGUIBILE DAL COMPUTER CHE LO RICONOSCE E LO ESEGUE OTTENENDO I RISULTATI.
 - **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. IL PROGRAMMATORE SCRIVE ANCHE IN QUESTO CASO IL CODICE SORGENTE E POI ATTRAVERSO UN INTERPRETE QUESTO VIENE INTERPRETATO E POI ESEGUITO.

Istruzioni

Sono fondamentali per poter eseguire e combinare i dati per creare l'algoritmo.
- operazioni [sequenziali] che vengono fatte una dopo l'altra come somma, divisione, moltiplicazioni.
Nei telefoni e computer moderni possono essere fatte anche operazioni in parallelo perché possiedono varie CPU ossia vari core.

- operazioni [condizionali] come IF in cui si arriva ad un punto in cui si deve scegliere se prendere una direzione o un'altra come ad esempio scegliere ad un certo punto se un numero è pari o dispari. Viene scelta quindi solo una delle due opzioni ossia quella che viene detta dalla macchina.

```
ex: var x = input
    if ( x divisibile per 2 ) * numero intero
        print ("pari" )
    else print ("dispari")
```

- operazioni [iterative]

- 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

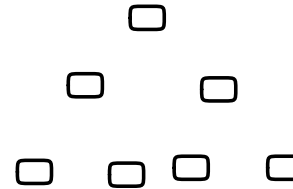
- Diagrammi a blocchi – flow chart
 - L'algoritmo viene rappresentato con un grafo orientato dove i nodi sono le istruzioni
 - Inizio e fine con ellissi
 - Rettangoli per le operazioni sequenziali (o blocchi)
 - Esagoni o rombi per condizioni
- Pseudo codice
 - L'algoritmo viene descritto usando l'approssimazione 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.




ordinare le stringhe significa sistamarle in ordine alfabetico, mentre ordinare una serie di numeri significa metterli in ordine di grandezza dal più piccolo al più grande.

il sorting è importante perchè ci permette di lavorare con algoritmi logaritmici che costano molto meno rispetto a quelli lineari. Essendo già ordinati è più semplice.

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 fase fondamentale che viene fatta per il progetto a cui si deve lavorare. Si valutano i requisiti per rigorizzare il progetto. In questo modo capiamo cosa davvero vogliamo ottenere dal progetto a cui andremo a lavorare.
 - Formalizzazione dell'idea di partenza, analisi costi e usabilità del prodotto atteso
- Progettazione progettazione vera e propria della applicazione, definendone la struttura del codice. Il progetto ovviamente sarà redatto dettagliatamente e per fare ciò utilizziamo lo pseudocodice.
 - Struttura complessiva del codice, definizione architetturale
 - Progetto di dettaglio, più vicino alla codifica ma usando pseudo codice o flow chart
- Sviluppo La verifica che il codice risponda alle specifiche che sono state date viene effettuata tramite unit test
 - Scrittura effettiva del codice, e verifica del suo funzionamento via **unit test** 
- Manutenzione
 - Modifica dei requisiti esistenti, bug fixing

Il programmatore è interessato a vedere se i requisiti sono stati rispettati e che il codice funziona realmente.
TDD = test driver development --> il programmatore stesso prima di scrivere la funzione bisogna scrivere i requisiti del test. ex: int count samanthas (string [] names).
Si valuta cosa può accadere normalmente ma anche ai casi negativi e ai casi limite in cui potrebbero succedere cose molto particolare.

Unit Test

FUNZIONE : blocco di codice con un nome a cui passo una serie di caratteri

- Verificano la correttezza di una singola “unità” di codice
 - Mostrano che i requisiti sono rispettati
- Verifica
 - Casi base (positivi e negativi)
 - Casi limite esempio: cosa succede se sono tutte samantha? cosa succede se samantha è il primo nome oppure nell'ultimo?
- 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

Se facciamo 10 volte il test allora questo deve dare tutte e 10 le volte lo stesso risultato. Questo è difficile da ottenere su input che non sono standard, come ad esempio se voglio testare una funzione che generi numeri casuali e questo non è possibile.

Se i test sono troppo complicati risulta più difficile scrivere il test piuttosto che il codice.

vogliamo che il nostro codice copra più o meno tutti i casi possibili.

Modello a cascata (waterfall)

In questa fase si stila un documento in cui si valutano i requisiti per la creazione del progetto, dando una stima realistica dei costi, del tempo necessario e di quando potrà vedere il cliente il prodotto finito.

Studio di fattibilità

Analisi dei requisiti

Progettazione

Manutenzione

Sviluppo

Unit test

Acceptance test

Integration test

Integrazione

Rilascio

se tutto quello che è stato fatto funziona allora si avrà il rilascio al cliente. C'è un altro test che viene fatto dal cliente prima del rilascio per vedere se tutto ciò che hanno fatto funziona. superato questo test fatto dal cliente e se il cliente è soddisfatto i programmatori "vengono pagati".

Esempio di una azienda di telefonia che vuole cambiare i programmi per la registrazione delle sue fatture. Nello studio di fattibilità il cliente e il programmatore parlano e devono cercare di capire domanda e offerta. Questo è il momento in cui ci si mette d'accordo mediando tra le richieste e le cose che si possono realmente fare, valutando costi e tempistiche.

In questa fase si progetta il programma che si deve fare nel tempo per avere una idea del risultato.

la fase di manutenzione risolve i problemi dopo che c'è il rilascio del progetto.

nello sviluppo il programmatore riceve degli input per poi dare degli output. ovviamente si lavora in un team, ogni programmatore esegue un piccolo lavoro e poi si mette insieme il tutto.

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