Informatica nei licei

Proposta di organizzazione per ore degli argomenti da trattare a cura di Andrea Aquino.
Revisionata da Andrea Melis e Sara Bergonzoni.

Premessa:

Considerando che le ore da dedicare all'informatica nei licei sono due alla settimana e considerando che ogni anno sono previste 33 settimane di studio divise in otto mesi si hanno a disposizione 8.25 ore mensili per l'insegnamento di tale materia. Considerando le festività ed il periodo prossimo alla fine di ogni anno accademico in cui gli studenti sono meno presenti è lecito approssimare le ore mensili da dedicare all'informatica ad 8.

La seguente è una proposta di organizzazione delle materie di studio (e degli argomenti relativi ad ogni materia) in accordo con la disposizione ministeriale.

Notazione:

[RIPARTIRE CARICO]: le ore etichettate in questo modo dovrebbero essere utilizzate per approfondire argomenti già trattati, per proporre provocazioni o aspetti particolarmente interessanti della materia informatica, per recuperare lezioni perse per qualche ragione, per organizzare esercitazioni in classe o in laboratorio o per ripartire argomenti particolarmente complessi su più ore di quante previste da questo piano.

[LABORATORIO]: le ore etichettate in questo modo dovrebbero essere utilizzate per esercitazioni mirate in laboratorio in cui gli studenti partecipano attivamente apprendendo in pratica gli argomenti trattati in modo teorico a lezione.

PRIMO ANNO:

[PRIMA PARTE]: Il primo anno si dedica all'introduzione alla materia informatica e all'approfondimento dei concetti chiave degli algoritmi dei calcolatori e delle strutture dati. Viene introdotta la sintassi e la semantica di un semplice linguaggio di programmazione: TurtleScript e di uno pseudocodice più espressivo per la soluzione di problemi di complessità crescente.

[SECONDA PARTE]: Segue un'introduzione all'architettura dei calcolatori partendo dalla loro storia ed evoluzione nel tempo analizzando poi in dettaglio l'architettura di Von Neumann con particolare riguardo per il funzionamento della CPU e della memoria principale e secondaria.

[TERZA PARTE]: E' previsto inoltre un accenno all'analisi dei documenti elettronici e dei linguaggi di markup, approfondendo le tecniche di sviluppo di pagine web statiche per mezzo di HTML e CSS.

PRIMA PARTE

2 ore (Introduzione): L'informatica, scienza che si occupa della trasmissione nello spazio e nel tempo di informazione prodotta in modo automatico. Cos'è l'informazione? Cosa sono i dati? Informatico come “problem solver” vivente, ottimizzatore e scienziato dell'informazione.

2 ore (Algoritmi): Concetto di algoritmo: chi li ha inventati, perchè sono così importanti nella vita reale. Ogni sequenza di passi elementari per svolgere una funzione è un algoritmo. Esistono infiniti algoritmi per risolvere un problema ma alcuni sono migliori di altri. Non tutti i problemi ammettono una soluzione algoritmica.

2 ore (Algoritmi): Introduzione della sintassi e della semantica di pseudocodice per l'implementazione teorica di algoritmi banali. Concetto di costante. Concetto di variabile inteso come contenitore in grado di contenere un valore. Aritmetica con variabili (operatori di somma, sottrazione, divisione intera, resto) con esercizi a fine lezione.

2 ore (Algoritmi): Potenziamento dello Pseudocodice con i costrutti condizionali (if), i costrutti di iterazione limitata (for) e illimitata (while e do-while). Agli studenti durante la lezione è richiesto di implementare l'operatore di moltiplicazione come iterazione di somme e di risolvere problemi banali (trasformare gradi celsius in fahrenheit e viceversa).
Compiti a casa: implementazione dell'algoritmo del calcolo del fattoriale in pseudocodice.

2 ore (Algoritmi): Introduzione di un primo semplice linguaggio di programmazione: TurtleScript. Cos'è e quali sono le primitive del linguaggio con particolare enfasi sulla possibilità di definire funzioni (intese come generalizzazione di sequenze di istruzioni a cui viene dato un nome, che restituiscono un risultato ed alle quali è possibile passare parametri). Geometria della tartaruga: perchè la tartaruga disegna "dall'interno". Perchè TurtleScript è meno espressivo dello pseudocodice visto nelle lezioni precedenti (manca di strutture dati). Disegno di un triangolo equilatero con la tartaruga. Applicazione dei costrutti base appresi in pseudocodice per muovere la tartaruga sulla canvas e relizzazione di semplici pattern per mezzo di un algoritmo.
Compiti a casa: riprodurre con la tartaruga un pattern assegnato dal professore che segua una logica rigorosa ma semplice. Eventuali esperimenti degli studenti sono apprezzati e commentati nella lezione successiva. Possibilmente il pattern proposto deve presentare più approcci possibili di risoluzione. Soluzioni diverse degli studenti vengono messe a confronto per risaltare il fatto che esistono infiniti algoritmi per la soluzione di un problema e che alcuni di questi sono migliori di altri. Hint: disegnare una casa .

2 ore (Algoritmi): La ricorsione come tecnica di programmazione. Come affrontare i problemi risolti fino a questo momento in modo ricorsivo (con particolare enfasi per gli algoritmi ricorsivi su numeri e liste). Funzioni ricorsive su numeri in TurtleScript.
Compiti a casa: scrivere una funzione ricorsiva in TurtleScript che realizzi un pattern molto semplice assegnato dal professore.

2 ore (Algoritmi): Introduzione di semplici strutture dati (liste) ed aggiunta di tali costrutti allo pseudocodice introdotto nelle prime lezioni. Algoritmo di ricerca del minimo elemento di una lista. Agli studenti è richiesto intervenire proponendo un algoritmo alternativo per calcolare il massimo elemento di una lista (basta scambiare il controllo elementoCorrente < minimoCorrente con elementoCorrente > massimoCorrente). Algoritmo di ordinamento per mezzo della ricerca del minimo.
Compiti a casa: realizzare un algoritmo in grado di restituire una lista che contenga solo i numeri pari di una lista data.

2 ore (Algoritmi): Concetto di coda e di pila. Realizzazione delle prime funzioni TurtleScript per il calcolo della funzione fibonacci in modo ricorsivo e nuovi esercizi su funzioni ricorsive su numeri e algoritmi in pseudocodice su code e pile.

2 ore (Algoritmi): Concetto di albero binario e ampliamento dello pseudocodice con i costrutti per navigare alberi binari. Semplici esercizi di ricerca ricorsiva di elementi in alberi binari.

2 ore (Algoritmi): Accenni agli alberi binari di ricerca: l'organizzazione dei dati migliora le prestazioni di ricerca. Come avvalersi di organizzazioni dei dati furbe per risolvere problemi in tempo più breve. Accenno ai grafi, come sono strutturati e con quale algoritmo è possibile visitarli (algoritmo di ricerca in profondità DFS).

2 ore: [RIPARTIRE CARICO]

SECONDA PARTE

2 ore (Architettura dei Calcolatori): Cenni storici dell'architettura dei calcolatori. Analytical Engine di Charles Babbage. Mutamento dell'importanza della figura del programmatore nel tempo: "Machine expensive, Programmers cheap: take machine busy!" verso "Programmers expensive, Machine cheap: take programmers busy!". Enigma, l'informatica come "arma" da guerra. Dalle schede perforate ai sistemi operativi.

2 ore (Architettura dei Calcolatori): Nel calcolatore tutto è rappresentato sotto forma di sequenza binaria ma lo spazio a disposizione è limitato. Il bit come particella atomica d'informazione. Introduzione alla codifica binaria. Aritmetica binaria, somma, sottrazione tra numeri binari.

2 ore (Architettura dei Calcolatori): moltiplicazione e divisione tra numeri binari. Operazioni di shift (aritmetico e logico) su numeri binari. Nuovi esercizi sull'aritmetica binaria.

2 ore (Architettura dei Calcolatori): Elettronica alla base dell'informatica. Le porte logiche ad alto livello: porta And, porta Or, porta Not, come combinarle tra loro per ottenere sommatori o altri componenti aggregati. Il transistor: come rappresentare il bit. Come utilizzare i transistors per realizzare porte logiche (Not, or, and).

2 ore (Architettura dei Calcolatori): Panoramica sul modello di calcolo di Von Neumann: CPU come unità di calcolo, BUS per la comunicazione tra le componenti hardware, memoria intesa come meccanismo per la memorizzazione di dati, rappresentati in codifica binaria, volatile (RAM) o permanente (disco). Il calcolatore è in grado di eseguire solo codice in un determinato linguaggio (detto linguaggio macchina), come fa dunque a eseguire un programma più complesso come quelli realizzati durante la prima parte del corso? Traduzione di linguaggi in linguaggio macchina (solo accenni, si intende trasmettere allo studente semplicemente il concetto che le unità di calcolo che saranno approfondite in seguito sono in grado di svolgere funzioni minimali ed il risultato di un algoritmo scritto in un linguaggio complesso è solo opera di una traduzione automatica verso tali funzioni base).

2 ore (Architettura dei Calcolatori): La CPU, i registri come forma base di memorizzazione. Registri diversi per funzioni diverse. Introduzione dell'ALU e del suo funzionamento. Una macchina è in grado di svolgere compiti estremamente semplici, agglomerati di istruzioni semplici costituiscono istruzioni complesse in grado di svolgere compiti più gravosi.

2 ore (Architettura dei Calcolatori): La memoria RAM, una memoria volatile. Cosa significa volatile? Spiegazione dell'organizzazione della ram in celle di pari dimensione "numerate". Perchè fare i calcoli in RAM e non sul disco: prestazioni maggiori. Provocazione: La RAM è indirizzata dai registri, la dimensione dei registri è un limite alla memoria indirizzabile.

2 ore (Architettura dei Calcolatori): Il disco come meccanismo di memorizzazione non volatile dei dati. Traccia, settore e cilindro. La testina come meccanismo di lettura/scrittura dei dati su disco.
Provocazione: Un disco è un supporto magnetico, l'analisi delle tracce magnetiche possono permettere di risalire a vecchi valori di determinate tracce: come operare la cancellazione permanente dei dati.

2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO

TERZA PARTE

2 ore (Struttura di Internet e Servizi): il web visto come un enorme grafo di pagine raggiungibili le une dalle altre. Panoramica dei protocolli utilizzati sul web per richiedere pagine o trasmettere dati ai server perchè siano elaborate: http, https (non si pretende alcun tipo di dettaglio implementativo di tali protocolli ma una semplice spiegazione del loro funzionamento ad alto livello). Concetto di URI e di URL, il metodo per identificare una pagina in modo univoco nella rete.

2 ore (Struttura di Internet e Servizi): Regolamentazione di Internet. La privacy, il copyright, il copyleft. Cenni di diritto di Internet e netiquette.

2 ore (Struttura di Internet e Servizi): Il linguaggio di markup HTML per esprimere ipertesti. Un documento è un albero! Organizzazione logica del testo sotto forma di paragrafi, capitoli, sotto-sezioni. Introduzione dei tag di uso più comune (html, head, body, title, p, div, span, h1, h2, ..., a, img) con particolare attenzione al tag "a" per definire link e àncore. Realizzazione delle prime pagine web, estremamente semplici. L'HTML non deve essere utilizzato per la presentazione di un ipertesto!

2 ore (Struttura di Internet e Servizi): il linguaggio XHTML. La struttura di XHTML è più rigida di quella di HTML. Esempi ed esercizi.

2 ore (Struttura di Internet e Servizi): Accessibilità ed usabilità. Come realizzare documenti HTML o XHTML accessibili ed usabili. Esempi ed esercizi.

2 ore (Struttura di Internet e Servizi): CSS-1, come aggiungere caratteristiche presentazionali ad un ipertesto HTML. Classi ed id. Attributi CSS più comuni (width, height, border, margin, padding, background, color).

2 ore (Struttura di Internet e Servizi): CSS-2, pseudo-classi (:hover, :first-child, :last-child, :visited). Altri tag HTML, panoramica dei tag attualmente utilizzati e dei tag deprecati. Storia dei frame e degli iframe, perchè oggi NON sono considerati una buona tecnica di produzione di pagine web.

2 ore (Documenti Elettronici): I font, la storia. Font con grazie e senza grazie. Monospaziati. Tecniche di buona organizzazione logica di un documento elettronico.

2 ore (Documenti Elettronici): codice ASCII ed UNICODE per la rappresentazione del testo. Accenni di codifica dei caratteri (con particolare riguardo per lo standard Utf-8).

2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO

SECONDO ANNO

[PRIMA PARTE]: Il secondo anno offre un ripasso della materia di algoritmi e l'introduzione di nuove ed interessanti strutture dati trattate nel dettaglio. Lo studente apprende in modo intuitivo il concetto di tempo di calcolo necessario affinchè un algoritmo termini e che l'organizzazione furba dei dati migliora le prestazioni dei propri algoritmi. Viene introdotto inoltre ad una classe di problemi risolubili in tempo esponenziale. Aggiunge costrutti utili allo pseudocodice appreso al primo anno ed apprende l'utilizzo di una versione aumentata di TurtleScript chiamata TS-AndryakLib.

[SECONDA PARTE]: In seguito lo studente apprende dettagli sul concetto di codice “open source”. Studia i sistemi operativi e tecniche di programmazione concorrente. Apprende il concetto di deadlock ed i principali algoritmi di gestione della memoria oltre che l'intuizione alla base dei file-system più diffusi.

PRIMA PARTE

2 ore (Algoritmi): Breve panoramica delle strutture dati viste al primo anno (liste, code, pile, alberi binari). Ordinamenti furbi su liste di numeri: Quicksort e Mergesort.
Compiti a casa: realizzare un algoritmo che presa in input una lista di numeri restituisca vero se esiste una somma tra due numeri della lista che da come intero K. Hint: ordinare la lista per ridurre il tempo di calcolo.

2 ore (Algoritmi): Ripetizione del concetto di albero binario di ricerca e relativo approfondimento. Come ottenere un albero binario di ricerca a partire da una lista di numeri. Introduzione agli alberi generalizzati (alberi ad N figli) implementati come nodi dotate di liste di figli. Semplici algoritmi di ricerca ricorsiva su alberi binari di ricerca e su alberi generalizzati.
Compiti a casa: Relizzare un algoritmo che restituisce il branching di nodo (campienza lista figli di un nodo) massimo su un albero generalizzato.

2 ore (Algoritmi): TurleScript con AndryakLib, come disegnare con la tartaruga "dall'esterno". Perchè AndryakLib risulta più potente di TurtleScript standard (permette di tracciare variabili d'ambiente che normalmente non sono a disposizione dell'utente (inclinazione della tartaruga, posizione della tartaruga e mette quindi a disposizione metodi per implementare funzioni che possano ritenere la tartaruga in una posizione standard ad ogni utilizzo senza comprometterne la posizione o la rotazione effettiva)

2 ore (Algoritmi): Applicazioni ricorsive della libreria TurtleScript per la genesi di semplici frattali (il triangolo di Sierpinski). Esempi di altri frattali più complessi: l'albero frattale e la curva di koch.

2 ore (Algoritmi): Ripetizione del concetto di grafo. Ripetizione dell'algoritmo di visita in profondità (DFS) e introduzione della visita in ampiezza (BSF). Introduzione della tecnica per cercare un cammino tra due nodi.

2 ore (Algoritmi): Gli archi dei grafi possono essere "pesati" e "orientati". Ogni grafo non pesato e non orientato può essere trasformato in un grafo pesato e orientato. Algoritmo di ricerca del cammino minimo tra nodi su grafi mediante l'algoritmo di Dijkstra.

2 ore (Algoritmi): Analisi dei grafi: cricca, insieme indipendente, copertura. Tutte e tre queste caratteristiche sono riconducibili le une alle altre, come fare?. Riducibilità dei problemi, a volte trovare un algoritmo furbo per risolvere un problema permette di avere in maniera quasi o del tutto automatica soluzioni per problemi ad esso riconducibili.

2 ore (Algoritmi): Concetto (trattato in modo estremamente semplice ed intuitivo) di complessità algoritmica nel caso pessimo. Quanti passi (in funzione della dimensione dell'input) farà al più il mio algoritmo per trovare la soluzione al problema.

2 ore (Algoritmi): Problemi con soluzione non polinomiale: le torri di hanoi, la leggenda, perchè al termine della risoluzione del problema presso il monastero di Hanoi plausibilmente finirà il mondo (64 dischi con lo spostamento di un disco al giorno costa 2^64 - 1 giorni di soluzione: 505390248594782 secoli, circa 3.5 milioni di volte l'età dell'universo). Altri problemi su grafi in NP: il commesso viaggiatore, ricerca di una cricca di dimensione K. Attualmente non esiste una soluzione efficiente al problema e trovarla significherebbe risolvere uno dei più grandi problemi aperti dell'informatica moderna: P = NP.

2 ore: [RIPARTIRE CARICO]
2 ore: [RIPARTIRE CARICO]

SECONDA PARTE

2 ore (Sistemi Operativi): La storia dei sistemi operativi, perchè un sistema operativo è così utile. La differenza tra i sistemi operativi proprietari ed i sistemi operativi open source. Perchè l'open source è ritenuto migliore dei prodotti proprietari a parità di funzionalità dalla comunità scientifica. Windows contro Linux contro Mac OS: esistono più sistemi operativi validi, libertà di scelta.
Hint: in questa lezione introduttiva si intende trasmettere allo studente il concetto chiave che non esiste un solo sistema operativo ma ve ne sono svariati e che alcuni sono presumibilmente migliori di altri sotto determinati aspetti. Si vuole spiegare inoltre quali sono i vantaggi di progettazione open source e perchè Linux riscuote così tanto successo nella comunità informatica (nessun costo di licenza, facile da installare e ridistribuire, mette a disposizione il controllo pressochè completo sulla macchina, vastamente diffuso (sebbene meno di Windows) ed ampiamente supportato da una comunità di sviluppatori aperta ed in perenne espansione)

2 ore (Sistemi Operativi): Il concetto di sistema operativo come processo demone (istanza di un programma in perenne esecuzione) che si preoccupa di gestire la memoria, la comunicazione con le periferiche e lo scheduling dei processi. Un processo è l'istanza esecutiva di un programma. Il sistema operativo altro non è che un processo che si preoccupa di gestire altri processi. Il sistema operativo: un programma fatto di decine di milioni di righe di codice.

2 ore (Sistemi Operativi): Il processo: una struttura dati che dispone di un certo spazio di memoria e di un codice (del programma di cui è istanza esecutiva). Il sistema operativo seleziona un processo e partendo dalla sua prima istruzione inizia ad eseguirlo. Monothread contro Multithreading: in principio i sistemi operativi gestivano un processo alla volta passando da uno all'altro rapidamente simulando un ambiente in cui tutti i programmi venivano eseguiti contemporaneamente (il SO esegue un pezzetto di un processo, lo blocca, ne prende un altro e ne esegue un pezzetto, lo blocca, ...). Ad oggi per aumentare le prestazioni si usano più processori per svolgere mansioni in contemporanea (come eseguire più processi contemporaneamente).

2 ore (Sistemi Operativi): La concorrenza. Cos'è e a cosa serve. Semplice problema: accesso concorrente ad un conto corrente. Perchè programmare in modo concorrente è più difficile che programmare normalmente. La velocità di esecuzione dei processi non è predicibile. Servono dei costrutti che permettano di sincronizzare i processi.
N.B: Questo corso liceale non tratterà le tecniche di "Test and Set" per atomicizzare le istruzioni ritenute troppo "a basso livello" per studenti liceali. Ci si limiterà ad introdurre il concetto di mutua esclusione ed i semafori. I monitor sono stati ritenuti troppo complessi per gli studenti liceali.

2 ore (Sistemi Operativi): Deadlock e Starvation: cosa sono e perchè vanno assolutamente evitati. Esempi di processi in starvation: la coda all'ufficio postale con i "furbi" che passano avanti. Come evitare il deadlock: mutua esclusione. Come evitare la starvation: turni.

2 ore (Sistemi Operativi): La mutua esclusione, a cosa serve e perchè è fondamentale nell'ambito della concorrenza. La mutua esclusione va usata con moderazione, solo le parti di codice che possono creare problemi di concorrenza andrebbero messe in mutua esclusione per aumentare le prestazioni d'esecuzione. Il semaforo: un meccanismo che permette di mettere in attesa i processi. Un processo chiede al semaforo se è possibile proseguire, qualora sia possibile il semaforo registra di aver concesso l'accesso ad un processo al blocco di codice successivo, se non è possibile il processo viene bloccato ed è costretto ad attendere finchè almeno uno dei processi cui è stato garantito l'accesso non affermi di aver finito le sue operazioni protette.

2 ore (Sistemi Operativi): I semafori, un utilizzo concreto mediante l'aggiunta di un costrutto ad hoc allo pseudocodice utilizzato nei primi due anni. Come risolvere il problema del conto corrente aggiungendo un blocco in mutua esclusione mediante l'utilizzo di un semaforo. Il problema dei filosofi a cena: N filosofi siedono ad un tavolo rotondo ed hanno a disposizione N forchette per mangiare spaghetti. Ogni filosofo può essere visto come un processo a se stante che segue i seguenti passi: 1) prende la forchetta alla propria destra se è disponibile altrimenti aspetta che sia disponibile. 2) prende la forchetta alla propria sinistra se è disponibile altrimenti aspetta che sia disponibile. 3) "mangia" per qualche secondo. 4) posa una forchetta alla propria destra. 5) posa l'altra forchetta alla propria sinistra. Cosa succede se tutti i filosofi prendono contemporaneamente la forchetta alla propria destra? per nessuno sarà disponibile la forchetta alla propria sinistra e quindi tutti attenderanno all'infinito che tale forchetta si liberi. Il problema del deadlock: attesa infinita causata da attesa mutuale per il rilascio di una o più risorse.
Hint: simulare il problema dei filosofi a cena organizzando la classe in cerchio e sfruttando delle penne per simulare le forchette. Gli studenti possono collaborare per cercare soluzioni al problema. (Soluzione: introdurre un filosofo mancino che inverte le operazioni 1 e 2)

2 ore (Sistemi Operativi): Lo scheduling. Cos'è e a cosa serve. Differenze: Schedule, Scheduling, Scheduler. Stati dei processi: Stop, Ready, Run. Gli scheduler FCFS (First Come First Served), vantaggi e svantaggi (facili da implementare ma in genere poco efficienti). Approccio allo scheduling di tipo FIFO mediante code (First In First Out).

2 ore (Sistemi Operativi): Come funzionano gli scheduler Round Robin? La presenza di quanti di tempo e l'uso delle code evita Starvation per quanto riguarda l'esecuzione dei processi. Quanto grande deve essere scelto il quanto di tempo? dipende! si usano euristiche.

2 ore (Sistemi Operativi): Algoritmo di Scheduling migliore in assoluto: SJF (Shortest Job First). Vantaggi e svantaggi: risolve sempre il problema nel modo migliore possibile ma non è implementabile in quanto il tempo di esecuzione di un processo non è predicibile. Tuttavia si usano euristiche per "indovinare" o approssimare al meglio il tempo d'esecuzione di un processo basandosi sulla media dei tempi di esecuzione passati.

2 ore (Sistemi Operativi): In cosa consiste la gestione della memoria principale. La memoria principale può contenere un certo numero di "pagine" di dati. Quando la memoria principale si satura è necessario decidere quale pagina buttar via per far posto per la nuova pagina richiesta. In questo entrano in gioco gli algoritmi di rimpiazzamento.

2 ore (Sistemi Operativi): LRU (Least Recently Used Algorithm). Politica di rimpiazzamento della memoria principale buttando via la pagina usata meno recentemente. Esempi ed esercizi in classe.

2 ore (Sistemi Operativi): MRU (Most Recently Used Algorithm). Politica di rimpiazzamento della memoria principale che butta via la pagina usata più recentemente. Esempi ed esercizi in classe.

2 ore (Sistemi Operativi): LFU (Least Frequently Used Algorithm). Politica di rimpiazzamento della memoria principale che butta via la pagina usata meno frequentemente. Esempi ed esercizi in classe.

2 ore (Sistemi Operativi): I file system, cosa sono e a cosa servono (esprimono l'organizzazione gerarchica dei file e delle directory). FAT (file allocation table), un primo semplice esempio di file system. Quali sono le sue caratteristiche. Utilizzato da MS-DOS. Nelle prime versioni non supportava le directory ad albero. I limiti di capacità (file e directory) FAT.

2 ore (Sistemi Operativi): File system Ext 2. Utilizzato da vecchie versioni di Linux. Cos'è un inode. Cos'è l'indirizzamento diretto e quello indiretto singolo e doppio. I limiti di capacità (file e directory) di Ext 2. Uno sguardo al futuro: accenni alla novità introdotte da Ext 3 e Ext 4.

2 ore (Sistemi Operativi): File system NTFS. Utilizzato dalle recenti versioni di Windows. Come funziona e quali sono i suoi limiti. Le specifiche NTFS non sono mai state rilasciate, NTFS è dunque chiuso e proprietario e meno affidabile rispetto Ext 3 ed Ext 4.

2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO

TERZO ANNO

[PRIMA PARTE]: Il terzo anno propone un ripasso della materia di algoritmi. Si dedica allo studio dei linguaggi di programmazione ed introduce un linguaggio di programmazione reale (e molto utilizzato) ma semplice: PHP.
In particolare gli studenti apprenderanno le metodologie d'uso di PHP4 e lo sostituiranno gradualmente allo pseudocodice utilizzato negli anni precedenti. E' consigliabile che gli studenti facciano il test gratuito messo a disposizione dal sito web www.w3school.com riguardo il linguaggio di programmazione PHP periodicamente per valutare le loro conoscenze. Oltretutto la guida al PHP presente sul medesimo sito è gratuita e ben fatta, potrebbe essere utilizzata dal docente come mezzo di insegnamento e dagli studenti come mezzo di approfondimento.

[SECONDA PARTE]: Durante la seconda parte del corso agli studenti sarà proposto il meta-linguaggio di markup XML ed i linguaggi standard per navigarne l'albero o per applicarvi trasformazioni (XPath, XSLT).

PRIMA PARTE

2 ore (Algoritmi): Introduzione della sintassi e della semantica di un linguaggio di programmazione (scripting) reale: PHP (PHP Hypertext Preprocessor). Somiglianze con lo pseudocodice visto negli anni precedenti. Le variabili e gli operatori. L'array come mezzo di rappresentazione delle liste. Un nuovo tipo di ciclo: foreach.
N.B: PHP è stato scelto perchè semplice, ricchissimo di funzioni built-in, ad alto livello (non richiede allocazione/deallocazione della memoria), turing completo.

2 ore (Algoritmi): PHP, un mezzo per testare i propri algoritmi. Introduzione degli array associativi. Panoramica delle principali funzioni built-in del linguaggio (echo, rand, funzioni su array, include). Agli studenti è richiesto implementare un algoritmo di ordinamento furbo a scelta (quicksort o mergesort) in PHP.

2 ore (Algoritmi): Localhost: un server privato in locale. Simula un server web dotato di una propria root sulla propria macchina. E' la chiave per eseguire script PHP. L'esito di un algoritmo PHP può essere visualizzato direttamente attraverso il browser. PHP è disponibile sia in ambiente Windows che Linux che Mac. Come installare l'interprete PHP ed eseguire codice PHP sulla propria macchina.
N.B: In questa lezione si vuole insegnare agli studenti l'occorrente per poter lavorare a casa propria in PHP facendo esperimenti e giocando con il linguaggio.

2 ore (Algoritmi): Laboratorio di PHP (esercizi ed approfondimenti). Risolvere problemi algoritmici che sono stati precedentemente trattati in pseudocodice con PHP.

2 ore (Algoritmi): PHP, le stringhe: non più entità a sè stanti ma "array di caratteri". Come stampare la lista dei caratteri che compongono una stringa in PHP. L'operatore punto (.) di concatenazione. Come leggere da file e scrivere su file in PHP.

2 ore (Algoritmi): Il linguaggio PHP può essere inglobato in codice HTML: PHP come mezzo per generare a sua volta codice HTML (o XHTML, o CSS). Automatizzare e rendere dinamico il contenuto di un sito web realizzato mediante PHP. Analisi di testo da file, calcolo delle ricorrenze delle lettere o delle parole in un testo.

2 ore (Algoritmi): Come leggere il contenuto di cartelle in PHP. Sfruttare funzioni ricorsive di tale linguaggio per navigare il file-system a profondità arbitraria stampando a video i nomi dei file ed altre informazioni. Attenzione ai possibili danni! PHP può cancellare un sottoalbero del file-system (del server locale) se non utilizzato con attenzione in questo modo.

2 ore (Algoritmi): GET e POST, due metodi standard per trasmettere dati ad un'applicazione php mediante dei form HTML. Come prelevare tali dati e come trattarli (funzione isset, array $_POST e $_GET). Cenni di sicurezza: non bisogna mai supporre che i dati che l'applicazione riceverà per mezzo di form sono quelli che ci si aspetta di ricevere. Per errore, per curiosità o per cattiveria gli utenti inseriranno dati errati che possono creare errori anche gravi nella visualizzazione dell'applicazione. Come difendersi: aumentare i controlli sui dati.

2 ore (Algoritmi): [LABORATORIO]

2 ore (Algoritmi): Concetto di macchina astratta, interprete e compilatore. Dai linguaggi di basso livello verso i linguaggi di alto livello. Storia della teoria dei linguaggi di programmazione.

2 ore (Algoritmi): Concetto di grammatica libera dal contesto (intesa come quadrupla: Set di simboli terminali, set di simboli non terminali, simbolo non terminale iniziale, insieme di produzioni). Descrizione accurata degli insiemi che costituiscono una grammatica, introduzione del simbolo terminale speciale "epsilon" (stringa vuota). Sintassi BNF, come esprimere una grammatica in poche e semplici mosse. La grammatica BNF espressa in BNF.

2 ore (Algoritmi): Linguaggio generato da una grammatica libera inteso come insieme di stringhe di soli terminali ottenibili mediante un numero (anche infinito) di applicazioni delle produzioni di una grammatica libera. Concetto di albero di derivazione di una stringa. Grammatiche ambigue: una grammatica che ammette due o più alberi di derivazione per la stessa stringa è ambigua. Esempio classico della grammatica ambigua del costrutto if-then-else.

2 ore (Algoritmi): Introduzione a Polygen, un sistema per generare testo su una grammatica libera a partire da una definizione BNF di questa. Semplici esempi in classe. Gli studenti possono intervenire e proporre variazioni alle grammatiche mostrate loro o proporne di nuove.

2 ore (Algoritmi): Automi a stati finiti (regolari). Cos'è uno stato, cos'è una transizione. Lo stato iniziale, gli insiemi di stati finali (o di accettazione). Gli automi a stati finiti visti come grafi i cui archi sono etichettati con simboli terminali. In che modo gli automi a stati finiti descrivono un linguaggio, si rendono così un metodo estremamente potente per determinare se una stringa appartiene al linguaggio da loro descritto o meno.

2 ore (Algoritmi): Le espressioni regolari, definizione e costrutti. La stella di kleene. Gli operatori derivati: +, ?: come ricavarli dagli operatori classici. Perchè le espressioni regolari sono un metodo estremamente compatto per descrivere piccoli linguaggi.

2 ore (Algoritmi): Equivalenza tra espressioni regolari ed automi regolari. Come trasformare un'espressione regolare nell'automa equivalente. Funzione preg_match in PHP: un metodo per determinare se una stringa appartiene o meno al linguaggio descritto da un'espressione regolare.

2 ore (Algoritmi): Automi a pila, cosa sono e in cosa si differenziano dagli automi comuni. Perchè gli automi a pila sono più espressivi degli automi standard: hanno "memoria" (sebbene limitata) delle transizioni passate. Esempi ed esercitazioni in classe.

2 ore (Algoritmi): Accenni alle grammatiche contestuali, cosa sono, perchè sono così complesse da trattare con un calcolatore.

2 ore (Algoritmi): //RIPARTIRE CARICO
2 ore (Algoritmi): //RIPARTIRE CARICO
2 ore (Algoritmi): //RIPARTIRE CARICO
2 ore (Algoritmi): //RIPARTIRE CARICO
2 ore (Algoritmi): //RIPARTIRE CARICO
2 ore (Algoritmi): //RIPARTIRE CARICO

SECONDA PARTE

2 ore (Documenti Elettronici): XML, un meta-linguaggio di markup. Perchè risulta così utile nell'organizzazione strutturata dei dati. Esempi di documenti XML per l'organizzazione dei dati relativi ad applicazioni e/o siti web (Hint: sito di una pizzeria con listato delle pizze in XML).

2 ore (Documenti Elettronici): XPath, un linguaggio per selezionare nodi all'interno di un documento XML. Le origini e perchè è così utile. Le basi: gli assi (cosa sono e come navigarli). Esercitazione su XPath in classe, agli studenti è richiesto collaborare per trovare espressioni XPath (banali) per recuperare particolari nodi (o set di nodi) all'interno di un documento XML.

2 ore (Documenti Elettronici): XPath, funzioni e metodi complessi.
N.B: si vuole trasmettere allo studente solo conoscenza approssimativa degli assi XPath e delle funzioni XPath più complesse, non dovrebbe essere lui richiesto di utilizzarle in casi difficili che possono confonderlo e ridurre la sua comprensione dell'argomento.

2 ore (Documenti Elettronici): XSLT, un metodo per trasformare un documento XML in un altro documento XML (o HTML, o XHTML). XSLT sfrutta XPath. Peculiarità: Possiede variabili ma queste NON variano! Non è in grado di esprimere cicli se non in modo ricorsivo. Il concetto di template in XSLT.

2 ore (Documenti Elettronici): XSLT, come funziona nel dettaglio. Esempi ed esercitazioni. Template ricorsivi ed iterativi. Il template identità: un template ricorsivo in grado di riprodurre un documento XML alla perfezione. Perchè il template identità è così comodo? (trasformazioni settoriali, es: cambio tutti i tag <b> in <i>).

2 ore (Documenti Elettronici): Come effettuare trasformazioni XSLT su documenti XML mediante PHP. Come passare variabili calcolate lato server da PHP ad una trasformazione XSLT. Svariati esempi ed esercizi in classe.
N.B: le trasformazioni XSLT vengono effettuate da una classe PHP5 (XSLT-transformer). Gli studenti devono momentaneamente prendere il set di comandi PHP che operano su tale classe ed il concetto di classe stessa come un costrutto a sè stante. Più tardi quando sarà loro introdotto il concetto di Classe e di linguaggio orientato agli oggetti il tutto risulterà più chiaro.

2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO

QUARTO ANNO

[PRIMA PARTE]: Gli studenti approfondiscono il concetto di linguaggio object oriented e le sue caratteristiche più importanti (classi, attributi, metodi, interfacce, astrazione, ereditarietà). Apprendono i meccanismi di programmazione ad oggetti attraverso il PHP5 di cui viene introdotta la sintassi.

[SECONDA PARTE]: Gli studenti apprendono l'importanza ed il funzionamento delle basi di dati. Imparano la differenza tra Data Definition Language e Data Manipulation Language, come esprimere interrogazioni in algebra relazione e in SQL standard e come vincolare tabelle di una base di dati.

PRIMA PARTE

2 ore (Algoritmi): introduzione alle classi ed ai linguaggi orientati agli oggetti. La storia. Concetto di programmazione orientata agli oggetti: cos'è una classe, cos'è un oggetto. I principali linguaggi orientati agli oggetti (c++, java). In che modo la separazione logica dei concetti in classi rende il codice più semplice ed ordinato.

2 ore (Algoritmi): Le classi nel dettaglio, come sono composte (attributi, metodi). Il costruttore di una classe. Il distruttore di una classe. Esempi ed esercizi sull'organizzazione dei dati mediante l'utilizzo di classi.

2 ore (Algoritmi): La visibilità dei metodi e degli attributi di una classe: public, static, private. In che modo agiscono sul codice e sulla visibilità tra classi.

2 ore (Algoritmi): L'ereditarietà e le sottoclassi. In che modo una classe eredita metodi ed attributi della classe che estende. Esempi ed esercizi in classe.

2 ore (Algoritmi): Le interfacce, cosa sono, a cosa servono. Esempi ed esercizi in classe.

2 ore (Algoritmi): L'astrazione. Classi astratte. Perchè le classi astratte non sono istanziabili, a cosa servono. In che modo definire una classe astratta migliora il codice e semplifica la logica di programmazione. Esempi ed esercizi in classe.

2 ore (Algoritmi): Introduzione al PHP5, estensione di PHP4 orientata agli oggetti. In che modo definire una classe in PHP5, come istanziare una classe, come definire i metodi di una classe. Esempi ed esercizi. (Es: potrebbe essere interessante che gli studenti creino un programma che gestisce liste di numeri mediante l'uso di classi).

2 ore (Algoritmi): PHP5, come definire un'interfaccia, una sottoclasse e una classe astratta. Esempi ed esercizi sull'argomento. Agli studenti è richiesto realizzare un piccolo programma che soddisfi semplici specifiche utilizzando i concetti di OOP appresi fino ad ora. (Es: un ristorante che serve piatti caldi e freddi. I piatti hanno un costo e un ingrediente base. I piatti caldi hanno un costo extra derivante dalla cottura. Dovrebbe essere implementato un metodo per il calcolo del prezzo di un piatto che restituisca il costo base del piatto più eventualmente il costo di cottura se si tratta di un piatto caldo).

2 ore (Algoritmi): [LABORATORIO]
2 ore (Algoritmi): [LABORATORIO]

2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO

SECONDA PARTE

2 ore (Basi di dati): introduzione alle basi di dati. Perchè è così importante lo storage permanente dei dati e perchè è necessario avere organizzazioni dei dati furbe che minimizzino il tempo di attesa per la ricerca ed il recupero di questi. Database contro File-System.

2 ore (Basi di dati): i database visti come collezioni di tabelle, come crearle, come manipolarle. Cos'è un DDL, cos'è un DML.

2 ore (Basi di dati): Introduzione all'algebra relazione, un approccio matematico al DML. Il concetto di superchiave e di chiave primaria. Costrutti di selezione e di proiezione. Esempi di interrogazioni in algebra relazione per ricavare set di dati da un insieme di dati.

2 ore (Basi di dati): Algebra relazionale, altri operatori: ridenominazione, join (natural, theta). Sufficienti per esprimere un numero molto grande di interrogazioni. Esempi e casi reali.

2 ore (Basi di dati): Come rappresentare l'assenza di informazione e/o la non conoscenza di informazione: i valori nulli. Perchè i valori nulli sono utili ma il loro abuso può portare ad avere tabelle di dati prive di senso.

2 ore (Basi di dati): SQL, sia DDL che DML. Qual'è la sua storia, quali sono i tipi di dato che può manipolare (interi, caratteri, testo, bit, date).

2 ore (Basi di dati): SQL in pratica: create table, alter table. Come creare una nuova tabella, come definirne gli attributi e come modificare una tabella esistente.

2 ore (Basi di dati): SQL in pratica: Come prelevare dati da un database. SELECT, clausola FROM e clausola WHERE. Il selettore globale *.

2 ore (Basi di dati): SQL e valori nulli, in che modo il database tratta selezioni su valori nulli. Come richiedere esplicitamente per mezzo di interrogazioni SQL campi con valori nulli.

2 ore (Basi di dati): SQL in pratica: Come inserire o modificare i dati presenti in un database. INSERT INTO, UPDATE.

2 ore (Basi di dati): SQL in pratica: Come raggruppare set di dati (clausola group by) o come ordinare le entries risultanti da un'interrogazione (order by).

2 ore (Basi di dati): SQL in pratica: gli operatori aggregati di SQL, avg, sum, count, max, min. La clausola having.

2 ore (Basi di dati): SQL in pratica: Interrogazioni annidate. Come esprimere interrogazioni molto complesse annidando interrogazioni più semplici.

2 ore (Basi di dati): Concetto di vincolo su una chiave. Perchè vincolare il database è assolutamente necessario (garantisce l'integrità dei dati). Vincoli SQL più comuni: check, foreign key.

2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO
2 ore: //RIPARTIRE CARICO

QUINTO ANNO

[PRIMA PARTE]: Gli studenti sono introdotti alle reti di calcolatori. Apprendono i livelli architetturali di rete ISO/OSI, le tecniche di trasmissione dei dati sul canale, il funzionamento dei principali hardware di rete, tecniche di controllo degli errori e le peculiarità dei protocolli di trasmissione dati più importanti.

[SECONDA PARTE]: Gli studenti apprendono il concetto di sicurezza informatica e le tecniche per realizzare comunicazioni private su canali pubblici. Imparano il funzionamento dei cifrari simmetrici ed asimmetrici, quali sono i principali attacchi attualmente utilizzati e come difendersi.

[TERZA PARTE]: Gli studenti apprendono il concetto di virtualità, la differenza tra simulazione ed emulazione ed il funzionamento (ad alto livello) delle reti virtuali e delle macchine virtuali.

PRIMA PARTE

2 ore (Reti di calcolatori): Introduzione alle architetture di rete. La rete come meccanismo per trasmissione dei dati nello spazio. Internet, cos'è e qual'è la sua storia.

2 ore (Reti di calcolatori): Livelli di astrazione ISO/OSI. I principali protocolli di comunicazione. Tecniche di codifica. Come i dati vengono trasmessi sul canale.

2 ore (Reti di calcolatori): Hardware di rete: bridge, switch, hub, ripetitori, router. Cosa sono, a cosa servono e come funzionano.

2 ore (Reti di calcolatori): Tecniche di controllo degli errori: bit di parità, checksum. Esempi ed esercizi in classe.

2 ore (Reti di calcolatori): Il livello "data link" analizzato nel dettaglio con particolare attenzione per il sottolivello MAC.

2 ore (Reti di calcolatori): La rete LAN, caratteristiche e funzionalità. Protocollo di comunicazione 802.1 (IEEE standard)

2 ore (Reti di calcolatori): Token Ring LAN, caratteristiche e funzionalità. Protocollo di comunicazione 802.5 (IEEE standard)

2 ore (Reti di calcolatori): Il protocollo UDP. Cos'è e in cosa consiste. Quali sono i suoi vantaggi, quali sono i suoi svantaggi. Casi d'uso reali: streaming online.

2 ore (Reti di calcolatori): Il protocollo TCP. Cos'è e in cosa consiste. Quali sono i suoi vantaggi, quali sono i suoi svantaggi. Casi d'uso reali: download dati da internet.

2 ore (Reti di calcolatori): Il protocollo IPv4. Cos'è e in cosa consiste. Quali sono i suoi vantaggi, quali sono i suoi svantaggi.

2 ore (Reti di calcolatori): Il protocollo IPv4 ormai non ha più senso di esistere, gli indirizzi IPv4 sono stati completamente esauriti. L'alternativa: il protocollo IPv6. Cos'è e in cosa consiste. Quali sono i suoi vantaggi, quali sono i suoi svantaggi.

2 ore (Reti di calcolatori): DNS, un database distribuito per la trasformazione di nomi di dominio in indirizzi IP (e viceversa). La sua storia ed il suo funzionamento.

2 ore: [RIPARTIRE CARICO]
2 ore: [RIPARTIRE CARICO]

SECONDA PARTE

2 ore (Sicurezza): Introduzione alla sicurezza informatica. Perchè è così importante. Casi reali di attacchi che hanno provocato danni per somme estremamente elevate di denaro. Definizione di "trustworthiness". Cosa minaccia il proprio sistema? Attackers per gioco, per cattiveria, per errore.

2 ore (Sicurezza): Comunicazione privata su un canale pubblico, come realizzarla? Tecniche di crittografia, la storia della crittografia e come questa sia stata usata nel corso dei secoli per celare informazioni al nemico. La sicurezza non deve risiedere nella segretezza dell'algoritmo ma deve risiedere nella segretezza di qualcosa di più piccolo e che possa cambiare facilmente (es: una password o una passphrase). Il cifrario di Cesare, cos'è e come funziona.

2 ore (Sicurezza): I cifrari a sostituzione, dal cifrario di Cesare ai cifrari a permutazione. Perchè il cifrario di Cesare è considerato un cifrario debole (bastano pochi tentativi per risalire al messaggio originale). In cosa consistono e come funzionano i cifrari a permutazione. Perchè i cifrari a permutazione sono considerati cifrari deboli (tecniche di analisi statistica delle occorrenze dei simboli per risalire al messaggio originale).

2 ore (Sicurezza): Protocolli a chiave simmetrica. In cosa consistono e come funzionano. Perchè sono così utilizzati ad oggi (velocità di crypting e decrypting). Quali sono i loro svantaggi principali? (necessitano della condivisione di un segreto tra mittente e destinatario per poter funzionare). DES, un meccanismo a chiave simmetrica per crittografare i messaggi. Come è strutturato e come funziona.

2 ore (Sicurezza): Protocolli a chiave asimmetrica, un metodo per comunicare in modo sicuro senza condividere un segreto in partenza. Quali sono i vantaggi e quali sono gli svantaggi di questo tipo di protocolli. RSA, un caso reale di protocollo di comunicazione sicura asimmetrica analizzato nel dettaglio. RSA, basi matematiche di funzionamento.

2 ore (Sicurezza): Come recuperare dati crittati anche se il segreto è andato perduto. Key Escrow, una tecnica per la divisione di un segreto in più pezzi.

2 ore (Sicurezza): Tipologie di attacco più comuni: Login Spoofing, Phishing, Key loggers, Sniffing. In cosa consistono e perchè ad oggi sono così pericolose.

2 ore (Sicurezza): Esostrutture, Intrusion detection, Virus detection. Come proteggere la propria macchina da attacchi ed intrusioni in maniera generale.

2 ore: [RIPARTIRE CARICO]
2 ore: [RIPARTIRE CARICO]

TERZA PARTE

2 ore (Sistemi virtuali): In cosa consiste la virtualità, la sua storia e perchè ad oggi è ritenuta così importante. Una scarpa è un martello virtuale. La differenza tra emulazione e simulazione. Cos'è una rete virtuale, cos'è una macchina virtuale.

2 ore (Sistemi virtuali): Le reti virtuali nel dettaglio, un modo per testare software su rete senza disporre di una rete.

2 ore (Sistemi virtuali): La virtualizzazione parziale: virtualizzare solo determinati aspetti di un concetto complesso, un metodo per guadagnare in prestazioni.

2 ore (Sistemi virtuali): La domotica, quando la virtualità si rende tramite per migliorare la vita di tutti i giorni. Concetto di base, proposte ed idee per come realizzarla.

2 ore (Sistemi virtuali): Approfondimento su un tema di virtualità a discrezione del docente.

2 ore: [RIPARTIRE CARICO]
2 ore: [RIPARTIRE CARICO]

2 ore (epilogo): L'informatica in Italia e l'informatica nel mondo. Situazione attuale e prospettive. L'università di Informatica in Italia e all'estero. In che modo seguire questa passione al meglio anche dopo il liceo.