Programmazione informatica triennio

Classe TERZA

Programmazione: dati strutturati

Competenze	Abilità	Conoscenze
• Individuare e utilizzare strutture dati appropriate per la soluzione di problemi • Modellare la soluzione di un problema (algoritmo) scomponendolo in sottoproblemi (sottoprogrammi) • Utilizzare i file per la conservazione permanente di dati	• Definire, scrivere, leggere, modificare array mono- e bi-dimensionali • Definire, scrivere, leggere, modificare stringhe • Definire, scrivere, leggere, modificare variabili record e vettori di record (tabelle) • Definire, scrivere, leggere, modificare file di testo e binari • Saper codificare semplici algoritmi attraverso un linguaggio di programmazione strutturato e controllarne l'esecuzione • Saper utilizzare un ambiente di programmazione integrato (IDE) per la creazione e la verifica di semplici programmi eseguibili dal computer o da altri dispositivi programmabili	• Array mono/bi-dimensionali • Algoritmi di ricerca e ordinamento sui vettori • Stringhe • Record e tabelle • File di testo e binari

Programmazione: sottoprogrammi

• Individuare e utilizzare strategie funzionali appropriate per la soluzione di problemi • Modellare la soluzione (algoritmo) di un problema complesso scomponendolo in sottoproblemi (sottoprogrammi) • Usare la programmazione imperativa per progettare e realizzare programmi complessi	• Definire algoritmi utilizzando i sottoprogrammi • Saper codificare semplici algoritmi attraverso un linguaggio di programmazione strutturato e controllarne l'esecuzione • Scegliere e utilizzare correttamente i parametri di un sottoprogramma, definendone la modalità d passaggio (per valore e per riferimento) • Saper utilizzare un ambiente di programmazione integrato (IDE) per la creazione e la verifica di semplici programmi eseguibili dal computer o da altri dispositivi programmabili • Saper creare una libreria di funzioni personalizzata (opz)	• Funzioni e procedure • Passaggio dei parametri per valore e per riferimento • Puntatori • Funzioni ricorsive • Librerie statiche e dinamiche (opz) • Ambiente locale, globale, di blocco e regole di visibilità • Record di attivazione • Memoria di programma, stack e heap • Librerie (opz)

Programmazione ad oggetti (OOP)

• Usare la programmazione orientata agli oggetti per progettare programmi complessi	• Definire una classe con attributi e metodi • Definire costruttori e distruttori di una classe • Saper modellare delle entità reali o astratte attraverso delle classi • Saper creare degli oggetti come istanze di classi • Classificare classi e relazioni tra di esse • Applicare i concetti di incapsulamento e *information hiding* • Riconoscere la gerarchia delle classi	• Classi e oggetti • Attributi e metodi • Incapsulamento: visibilità di attributi e metodi • Costruttore/distruttore di una classe • Ereditarietà • Polimorfismo • Overriding • Overloading

	• Rappresentare classi e oggetti mediante diagrammi UML	• Diagrammi UML

Linguaggi del Web (HTML, CSS, Javascript)

• Progettare e realizzare risorse web ipertestuali dinamiche	• Creare semplici moduli HTML per l'inserimento di dati • Formattare i contenuti delle pagine HTML utilizzando le proprietà CSS • Definire e attivare uno script Javascript all'interno di una pagina HTML • Utilizzare Javascript per la validazione dei moduli HTML	• Moduli HTML: caselle di testo, pulsanti opzioni e scelta multipla, elenchi a discesa • Sintassi base del CSS • Proprietà CSS per la formattazione del testo e della pagina HTML • Sintassi base del Javascript

Classe QUARTA

Archivi su file (linguaggio C)

Competenze	Abilità	Conoscenze
• Gestire un archivio dati memorizzato su file	• Gestire dati complessi (record) in un file binario attraverso operazioni di lettura, scrittura, modifica, cancellazione	• File binari: operazioni di apertura/chiusura, lettura/scrittura, accesso diretto

Database

• Utilizzare lo schema concettuale dei dati E-R • Utilizzare il modello logico dei dati • Utilizzare gli operatori relazionali • Rispettare le regole di integrità	• Progettare basi di dati relazionali • Individuare entità e relazioni all'interno di una realtà di interesse di media difficoltà • Realizzare un modello concettuale e logico di una base di dati relazionale • Realizzare e gestire una base di dati su DBMS (es. mySQL) utilizzando una interfaccia grafica (es. myPhpAdmin)	• Confronto database/archivi su file • Struttura ANSI/SPARC • Caratteristiche funzioni di un DBMS • Fasi di progettazione di database relazionali • Progettazione concettuale • Entità, attributi, chiavi • Relazioni: definizione, rappresentazione, grado, cardinalità, direzione esistenza • Ristrutturazione dello schema concettuale • Modello logico relazionale: rappresentazione, regole di traduzione del modello concettuale • Chiave primaria/esterna • Integrità referenziale, regole di inserzione, cancellazione e modifica dei record • Operatori relazionali: proiezione, selezione, unione, differenza, intersezione, prodotto, congiunzione
• Inserire, ricercare e presentare dati contenuti in una base di dati	• Definire una interrogazione (query) ad una o più tabelle di una base di dati utilizzando il linguaggio SQL • Definire il codice SQL per l'inserimento di dati in una tabella di una base di dati	• Linguaggio SQL • DML: operatori di confronto, aritmetici, logici • Operazioni di selezione, proiezione e congiunzione tra tabelle • Operatori di aggregazione COUNT, SUM, MIN, MAX • Clausola GROUP BY, la condizione HAVING

Database in rete (linguaggio PHP)

• Gestire un database in rete • Creare e progettare risorse web dinamiche	• Creare pagine PHP per la connessione e accesso in lettura/scrittura ad un database in rete • Creare pagine web dinamiche utilizzando dati prelevati da un database in rete	• Linguaggio PHP (sintassi base) • Connessione e accesso in lettura/scrittura ad un database MySQL su server

Classe QUINTA

Reti informatiche

Competenze	Abilità	Conoscenze
• Classificare le reti in base alla topologia e ai mezzi trasmissivi utilizzati • Individuare i diversi dispositivi di rete • Individuare il formato del messaggio HTTP • Gestire il trasferimento di file utilizzando i comandi FTP	• Riconoscere le funzioni in base ai diversi livelli protocollari • Confrontare il modello ISO/OSI con la pila TCP/IP • Scomporre una rete in sottoreti • Utilizzare le principali applicazioni di rete	• Elementi fondamentali di una rete • Topologie fisiche di rete • Techiche di trasmissione: acccesso al canale, multiplexing, switching • Modello ISO/OSI e protocolli TCP/IP • Struttura e classi degli indirizzi IP, indirizzi pubblici/privati, statici/dinamici • Subnetting • Applicazioni di rete • Porta e socket • Architetture client/server e peer-to-peer • Architettura del Web • Protocolli applicativi HTTP e FTP • Servizi email e DNS

Elementi di crittografia

• Adottare e utilizzare consapevolmente i sistemi di protezione e trasmissione sicura dei dati	• Curare i dati considerando i principali aspetti della sicurezza informatica • Trasporre in codice di programmazione semplici metodi di cifratura per trasposizione e sostituzione • Utilizzare con consapevolezza gli strumenti di identità digitale	• Sicurezza fisica e logica dei dati • Backup/restore, cloud • Principi di sicurezza informatica • Crittografia simmetrica/asimmetrica • Analisi delle frequenze • Algoritmi crittografici • Firma digitale, PEC, SPID • Blockchain

Calcolo numerico

• Conoscere e utilizzare algoritmi per il calcolo numerico • Valutare la bontà di un generatore di sequenze pseudocasuali	Implementare, attraverso un linguaggio di programmazione: • il metodo dei rettangoli, dei trapezi e di Cavalieri-Simpson • lo sviluppo in serie di Eulero per il calcolo approssimato di *e* • lo sviluppo in serie di Taylor e Mc Laurin per il calcolo approssimato del seno di un angolo • l'algoritmo LCG per la generazione di numeri pseudocasuali • il metodo Monte Carlo per il calcolo delle aree e di π	• Basi del calcolo numerico • Algoritmi per la generazione di sequenze di numeri pseudocasuali • Algoritmi per il calcolo approssimato delle costanti π e *e* • Algoritmi per il calcolo approssimato delle aree • Algoritmi per il concetti fondamentali sui metodi di discretizzazione • Algoritmo LCG

Complessità degli algoritmi

• Individuare i parametri di	• Calcolare la complessità temporale di un algoritmo	• Qualità di un algortimo • Complessità asintotica di un

qualità di un algoritmo • Saper scegliere l'algoritmo più adeguato al problema • Individuare i problemi non computabili	• Classificare gli algoritmi in base alle classi di complessità • Definire il polinomio caratteristico di un codice • Determinare la complessità mediante l'istruzione dominante	algoritmo • Notazione O-grande (Ω-grande, Θ-grande) • Problemi, computabili/non computabili, trattabili/intrattabili • Classi di complessità: P, NP e NPC