Allegato B2

Quadro degli obiettivi formativi specifici e delle propedeuticità
Corso di Laurea in Informatica
Rau, art. 12

Insegnamento	Settore Scientifico Disciplin.	Obiettivi formativi specifici (ITA)	Specific educational objectives (ENG)	Propedeuticità

Algoritmi e strutture dati e laboratorio	INF/01	Indice:
		• ***Fondamenti***
		  ○ *Ruolo degli algoritmi*
		  ○ *Progettazione di algoritmi*
		  ○ *Correttezza degli algoritmi*
		  ○ *Complessità computazionale*
		• ***Strutture Dati di Base e relativi Algoritmi***
		  ○ *Vettori, Heap, Liste, Pile, Code, Alberi Binari, Tabelle di Hash, e varianti.*
		  ○ *Algoritmi per l'ordinamento, la ricerca, l'inserimento e la cancellazione in strutture dati di base*
		  ○ *Dimostrazione della correttezza di tali algoritmi*
		  ○ *Valutazione della complessità computazionale di tali algoritmi*
		• ***Strutture Dati Avanzate, Algoritmi e Tecniche Avanzate di progettazione***
		  ○ *Bilanciamento di alberi binari con algoritmi di ricerca/inserimento/cancellazione*
		  ○ *B-alberi con algoritmi di ricerca/inserimento/cancellazione*
			Index:
			• ***Basics***
			  ○ *The role of algorithms*
			  ○ *Algorithm Design*
			  ○ *Correctness*
			  ○ *Computational Complexity*
			• ***Basic Data Structures and Algorithms***
			  ○ *Arrays, Heaps, Lists, Stacks, Queues, Binary Trees, Hash Tables, and similars.*
			  ○ *Algorithms for sorting, searching, insertion, deletion over basic data structures*
			  ○ *Proofs of correctness*
			  ○ *Computational complexiety evaluations*
			• ***Advanced Data Structures and Design Techniques***
			  ○ *Balanced binary trees and search/insert/delete algorithms*
			  ○ *B-trees and search/insert/delete algorithms*
			  ○ *Data structures for disjoint sets*
				VINCOLANTE:
				Programmazione e laboratorio
				
				CONSIGLIATE:
				Analisi Matematica, Matematica discreta

		○ *Strutture Dati per insiemi disgiunti con operazioni Make, Union, Find e relative euristiche* ○ *Grafi e algoritmi di base sui grafi* ○ *Alberi minimi di copertura per grafi pesati* ○ *Cammini minimi in grafi pesati* ○ *Dimostrazione della correttezza e valutazione della complessità di tali algoritmi* Lo/la studente/essa dovrà: **Capacità relative alle discipline** **1.1**. *Conoscenza e capacità di comprensione* Conoscere i concetti base relativi alle strutture dati statiche e dinamiche e alla progettazione di algoritmi Conoscere un linguaggio formale per la descrizione degli algoritmi Conoscere gli strumenti matematici per la valutazione della complessità computazionale Conoscere le strutture dati e gli algoritmi presentati a lezione e le relative complessità spaziali e temporali **1.2** *Capacità di applicare conoscenza e comprensione* Saper implementare e testare gli algoritmi visti a lezione nei linguaggi di programmazione più comuni Saper utilizzare le strutture dati e gli algoritmi presentati nel corso per progettare nuovi algoritmi in grado di risolvere specifici problemi Saper dimostrare la correttezza e valutare la complessità computazionale di tali algoritmi **Capacità trasversali / soft skills**	○ *Graphs and basic graph algorithms* ○ *Minimum Spanning Trees* ○ *Shortest paths problems/algorithms* ○ *Correctness and Complexity of the above listed algorithms* The student should be able to: **Sector-specific skills** **1.1**. *Knowledge and understanding* Manage basic concepts of static and dynamic data structures and design of algorithms. Know a formal language for describing algorithms. Evaluate computational complexities. Know the data structures and algorithms presented during the course together with their spatial and temporal complexities. **1.2** *Applying knowledge and understanding* Implement and test algorithms seen in the course exploiting the most common programming languages. Use the data structures and the algorithms presented in the course to design new algorithms that can solve specific problems. Prove the correctness and evaluate the computational complexities of such algorithms **Cross-sectoral skills/soft skills** **2.1** *Making judgements* Determine which data structures, algorithms and design techniques use to provide efficient solutions to a given algorithmic problem. Choose the most computationally efficient solution. Combine/extend existing algorithms to solve a	

		2.1 *Autonomia di giudizio* Saper valutare quali strutture dati, quali algoritmi e quali tecniche di progettazione utilizzare per fornire una soluzione efficiente ad un dato problema algoritmico Saper scegliere tra più possibili soluzioni quella computazionalmente più efficiente Saper combinare/estendere algoritmi esistenti per risolvere uno specifico problema Saper dimostrare o confutare la correttezza e valutare la complessità computazionale di algoritmi da lui/lei progettati o reperiti in letteratura Saper selezionare il linguaggio di programmazione e le librerie più adatte per l'implementazione degli algoritmi Saper testare le implementazioni. **2.2** *Abilità comunicative.* Essere in grado di motivare, le scelte effettuate in termini di strutture dati e algoritmi per la risoluzione di un dato problema. Essere in grado di reperire le eventuali informazioni aggiuntive necessarie per la progettazione di un algoritmo efficiente. **2.3** *Capacità di apprendimento* Saper reperire ed utilizzare risorse informatiche e scientifiche per l'approfondimento autonomo delle tematiche studiate a lezione Saper affrontare lo studio di nuove problematiche di natura algoritmica Saper individuare nuovi strumenti per lo sviluppo e l'implementazione di strutture dati e algoritmi	specific problem. Demonstrate or refute the correctness and evaluate the computational complexity of algorithms. Select the programming language and the most suitable libraries for implementing an algorithm. Test implementations. **2.2** *Communication skills.* Motivate, make choices in terms of data structures and algorithms for solving a given problem. Find any additional information needed to design an efficient algorithm. **2.3** *Learning skills* Find and exploit scientific resources for the in-depth study of topics studied in the course. Deal with new problems of algorithmic nature. Identify new tools for developing data structures and algorithms.	
Analisi matematica	MAT/05	Il corso di Analisi Matematica vuole fornire una solida preparazione di base nel calcolo differenziale e integrale in una variabile. La	The Calculus course gives a solid grounding in the differential and integral calculus in one variable. There are two purposes: familiarizing	

		finalità è duplice: familiarizzare il discente con il metodo dimostrativo in generale, e nel contempo fornire nozioni senza le quali si è tagliati fuori dalla comprensione della statistica, dell'analisi dei dati numerici, nonché di buona parte dell'informatica teorica. La teoria viene presentata senza rinunciare a un certo livello di rigore formale, modulando il passo con la progressiva familiarità che gli studenti acquisiscono con il metodo logico-deduttivo. Enfasi viene data al significato geometrico intuitivo dei concetti di limite, derivata e integrale. **Argomenti**: nozioni di logica, gli insiemi numerici con operazioni e ordinamento, l'induzione matematica, limiti, derivate, integrali in una variabile. **Capacità relative alle discipline** *Conoscenza e capacità di comprensione*: conoscere il significato geometrico e algebrico di limiti, derivate, integrali. *Conoscenza e capacità di comprensione applicate*: saper manipolare in modo logicamente corretto espressioni contenenti limiti, derivate, integrali e saperle usare per risolvere problemi. **Capacità trasversali/soft skills** *Autonomia di giudizio*: saper collegare fra loro idee algebriche, geometriche e computazionali. *Abilità comunicative*: saper scrivere in modo comprensibile la risoluzione di un problema matematico, e saper dimostrare oralmente un teorema. *Capacità di apprendimento*:	the student with the demonstrative method in general, and, at the same time, teach the notions that are a prerequisite for understanding statistics, numerical data analysis and much of theoretical computer science. The theory is explained without forfeiting some level of formal rigour, gradually accompanying the student in the discovery of the logical-deductive method. Emphasis is given to the intuitive geometric meaning of the notions of limit, derivative and integral. **Topics**: Notions of logic, numerical sets with algebraic operations and ordering, mathematical induction, limit, derivative, integral in one variable. **Sector-specific skills** *Knowledge and understanding*: The algebraic and geometrical meaning of limit, derivative, integral. *Applying knowledge and understanding*: Being able to manipulate in a logically correct way expressions involving limits, derivative, integral, and using them to solve problems. **Cross-sectoral skills/soft skills** *Making judgements*: Being able to make useful connections between algebraic, geometric and computational ideas. *Communication skills*: Being able to write a report on how to solve a mathematical problem, and to prove a theorem orally. *Learning skills*:	

		saper leggere un testo scientifico che usa il metodo dimostrativo e gli strumenti di base dell'analisi matematica.	Being able to read a scientific text that uses the demonstrative method and the basic notions of calculus.	
Architettura degli elaboratori e laboratorio	INF/01	Scopo del corso è descrivere la struttura e il funzionamento dei calcolatori nelle loro diverse componenti, illustrando le principali tecnologie utilizzate. Il percorso didattico inizia con lo studio delle parti elementari che costituiscono un computer e considera componenti sempre più complesse fino ad arrivare allo studio di architetture complete di calcolatori. Argomenti trattati sono: la progettazione di semplici circuiti logici, le codifiche hardware dell'informazione, la struttura del processore, il collegamento e la comunicazione tra periferiche e processore, il tema della memoria (cache, centrale, di massa, virtuale), le architetture parallele (pipelining, superscalari, sistemi multiprocessore), la programmazione in linguaggio assembly. Il corso è accompagnato da una parte di laboratorio in cui lo studente si esercita nella progettazione di circuiti logici e nella programmazione in linguaggio macchina. **Capacità relative alle discipline** 1.1 *Conoscenza e capacità di comprensione*: dopo aver superato l'esame si ritiene che lo studente sia in grado di comprendere il funzionamento dell'hardware e conoscere i fattori che influenzano le prestazioni dei calcolatori. 1.2 *Conoscenza e capacità di comprensione applicate*: l'attività di laboratorio permette allo studente di impratichirsi con la progettazione a basso livello, sia di semplici sistemi digitali, che di	The course aims at describing the structure and operations of the diverse components forming a computer, by illustrating the most popular technologies. The course starts with an analysis of the basic elements of a computer, and then it considers components that are progressively more complex until arriving at the study of complete computer architectures. The following arguments will be dealt with: design of simple logic circuits, hardware coding of the information, structure of a processor, communication between a processor and its peripherals, memory (central, cache, mass, virtual), parallel architectures (pipelining, superscalar, multi-processor), programming in assembly language. The lectures include laboratory sessions where students will exercise on logic circuit design and machine language programming. **Sector-specific skills** 1.1 *Knowledge and comprehension skills*: past the exam, students will understand how computer hardware works; furthermore, they will know the main factors that influence the performance of a computer. 1.2 *Knowledge and comprehension applied skills*: the laboratory activity will allow students to get used with the low-level design of both simple digital systems and programs in	

		programmi scritti in linguaggio macchina, ossia eseguibili direttamente dal calcolatore. **Capacità trasversali/soft skills** 2.1 *Autonomia di giudizio*: lo studente acquisisce gli strumenti per poter valutare e scegliere in maniera ponderata l'architettura hardware da utilizzare. 2.2 *Abilità comunicative*: lo studente impara l'esatto significato dei vari termini usati per descrivere le caratteristiche e le specifiche di un calcolatore ed usarle in maniera appropriata. 2.3 *Capacità di apprendimento*: il corso mira a fornire allo studente un quadro concettuale sufficientemente rigoroso completo da permettergli di affrontare con sicurezza lo studio dettagliato delle varie componenti di un calcolatore.	machine language, i.e. directly executable by the computer. **Cross-sectoral skills/soft skills** 2.1 *Making judgements*: students will learn tools for evaluating with ponder and choosing a hardware architecture. 2.2 *Communication skills*: students will learn the exact meaning of the different words used for properly describing characteristics and specifications of a computer. 2.3 *Learning skills*: the course aims at providing a conceptual framework, precise enough to secure a detail study of the diverse components finding place in a computer.	
Basi di dati e laboratorio	INF/01	Obiettivo fondamentale del corso è l'acquisizione dei concetti, degli strumenti e delle metodologie fondamentali nel campo delle basi di dati, con particolare attenzione ai modelli (concettuale, logico e fisico), ai linguaggi (di definizione, di aggiornamento e di interrogazione) e all'architettura dei sistemi per basi di dati. Vengono fornite le nozioni fondamentali inerenti ai linguaggi per la definizione, l'interrogazione e l'aggiornamento dei dati (algebra relazionale, calcolo relazionale, SQL). Vengono inoltre forniti elementi di progettazione concettuale (raccolta e analisi dei requisiti, costruzione di modelli Entità/Relazioni), logica (ristrutturazione di schemi concettuali, trasformazione di schemi concettuali in schemi logici, normalizzazione dei dati) e fisica (strutture di indicizzazione). Infine, viene introdotta la nozione di transazione e	The overall aim of the course is to learn the concepts, tools and fundamental methodologies in the realm of database management, with a special emphasis on conceptual, logical, and physical modeling, on data definition and manipulation languages, and on database system architecture. The fundamental notions regarding definition and query languages are explained, as well as elements of conceptual modeling (from requirement elicitation and analysis to the construction of Entity-Relationship models), logical modeling (including schema restructuring, translation into the Relational Model, and data normalization), and physical modeling (indexing). Finally, the student will learn about transaction management and will acquire an understanding of the main	CONSIGLIATE: Algoritmi e strutture dati e laboratorio, Sistemi operativi e laboratorio

		vengono analizzate le componenti principali di un sistema di gestione dei dati. Dopo aver superato l'esame si ritiene che lo studente sia in grado di formalizzare in un linguaggio relazionale operazioni di definizione e manipolazione dei dati espresse in linguaggio naturale e di progettare semplici basi di dati a livello concettuale (costruzione di schemi Entità/Relazioni a partire da insiemi di requisiti espressi in linguaggio naturale), logico (trasformazione di schemi Entità/Relazioni in schemi relazionali, formalizzazione della semantica intesa delle relazioni tramite dipendenze funzionali, normalizzazione di schemi relazionali) e fisico (definizione degli opportuni indici). **Capacità relative alle discipline** **1.1.** *Conoscenza e capacità di comprensione* • Concetto di "modello dei dati". • Modelli per la progettazione concettuale di una base di dati. • Il modello relazionale. • Linguaggi formali per la definizione e la manipolazione dei dati: algebra relazionale e calcolo relazionale. • Elementi architetturali di un sistema di gestione dei dati. • Politiche e meccanismi per la gestione concorrente delle transazioni. • Algoritmi e strutture dati per l'indicizzazione efficiente dei dati. **1.2.** *Conoscenza e capacità di comprensione applicate* • Elementi di progettazione concettuale: analisi e raccolta dei requisiti, diagrammi Entità/Relazione.	components of a database management system. After this course, the student will be able to formalize into a relational language operations on data informally expressed in a natural language. The student will also be able to understand and develop conceptual database designs (from requirements to E-R diagrams), logical database designs (from the translation of E-R models into relational schemas to the formalization of relational semantics through functional dependencies and schema normalization), and physical database designs (definition of index data structures). **Sector-specific skills** **1.1.** *Knowledge and understanding* • Notion of "data model". • Models for conceptual database design. • The Relational Model. • Formal languages for data definition and manipulation: Relational Algebra and Relational Calculus. • Architecture of a database management system. • Policies and mechanisms for concurrent transaction management. • Algorithms and data structures for efficient data indexing. **1.2.** A*pplying knowledge and understanding* • Elements of conceptual design: requirement analysis, Entity/Relationship diagrams. • Logical and physical database design. • The SQL language.	

		• Progettazione logica e fisica di una base di dati. • Il linguaggio SQL. • Procedure memorizzate, funzioni definite dall'utente e trigger. • Casi di studio: i sistemi di gestione dei dati PostgreSQL e MySQL. **Capacità trasversali / soft skills** **2.1.** *Autonomia di giudizio* • Analizzare e formalizzare i requisiti fondamentali di un sistema informativo. • Confrontare soluzioni diverse di un problema di gestione dei dati. • Formulare in modo corretto interrogazioni complesse. • Valutare l'opportunità o meno di usare strutture d'indicizzazione per l'ottimizzazione delle interrogazioni. **2.2.** *Abilità comunicative* • Interagire con i soggetti direttamente o indirettamente coinvolti in un progetto di gestione dei dati. • Motivare le scelte poste in atto in tutte le fasi della progettazione di un sistema informativo. **2.3.** *Capacità di apprendimento* • Apprendere in modo autonomo il funzionamento di ulteriori software di gestione di gestione dei dati. • Approfondire ulteriori tematiche inerenti alle basi dei dati, quali, ad esempio, le basi di dati XML, i data warehouse, la sicurezza e protezione dei dati. • Affrontare le problematiche di gestione dei dati in applicazioni specifiche, come	• Stored procedures, user-defined functions, and triggers. • Case studies: PostgreSQL and MySQL. **Cross-sectoral skills/soft skills** **2.1.** *Making judgments* • Analyze and formalize the fundamental requirements of an information system. • Compare different solutions to a data management problem. • Correctly formulate complex queries. • Evaluate the opportunity of using indexes for query optimization. **2.2.** *Communication skills* • Interact with (technical and non-technical) stakeholders involved in a data management project. • Convincingly argue about the decisions taken during all the phases of the design of an information system. **2.3.** *Learning skills* • Independently learn how to use other database management software. • Delve into further topics in data management, such as XML databases, data warehouses, data security and privacy. • Tackle data management issues in specific applications, such as biological or spatio-temporal databases.	

		ad esempio, le basi di dati biologici o spazio-temporali.		
Calcolo delle probabilità e statistica	SECS-S/01	L'insegnamento introduce i concetti fondamentali del Calcolo delle Probabilità e della Statistica, quale strumentazione di base per lo studio dei fenomeni aleatori, e fornisce le competenze necessarie per impostare e condurre le più semplici analisi statistiche. Offre, inoltre, chiavi di lettura per un approfondimento individuale delle problematiche legate alla Statistica, con particolare riferimento alle applicazioni. I punti principali toccati sono: 1. Introduzione al Calcolo delle Probabilità e risultati di base 2. Variabili casuali, unidimensionali e multidimensionali, discrete e continue 3. Momenti e funzione generatrice dei momenti 4. Modelli probabilistici notevoli 5. Convergenza in probabilità e in distribuzione 6. Introduzione all'Inferenza Statistica **Capacità relative alle discipline** **1.1**. *Conoscenza e capacità di comprensione* Lo studente deve avere acquisito le conoscenze di base: della probabilità e probabilità condizionale (teorema di Bayes); delle variabili casuali (sapendo ben distinguere leggi discrete e leggi continue); del valore atteso, della varianza, degli ulteriori momenti, sapendo anche utilizzare la funzione generatrice dei momenti; delle principali leggi di probabilità univariate; dei modi di convergenza di successioni di variabili casuali (con i teoremi limite); delle procedure di inferenza statistica nei più semplici modelli statistici parametrici di campionamento casuale.	This course gives an elementary introduction to Probability with a view towards Statistics. The main topics are 1. Elementary probability 2. Random variables, unidimensional and multidimensional, discrete and continuous 3. Moments and moment generating function 4. Special univariate distributions 5. Modes of convergence, LLN and CLT 6. Introduction to statistical inference **Sector-specific skills** **1.1**. *Conoscenza e capacità di comprensione* The student should acquire basic notions of: probability and conditional probability (Bayes' theorem); random variables (with secure distinction between discrete and continuous distributions); expectation, variance, and further moments, being also able to use the moment generating function; the main univariate probability distributions; the modes of convergence of sequences of random variables (and limit theorems); the usual procedures of statistical inference in the simples models of random sampling. **1.2** *Capacità di applicare conoscenza e comprensione* The student should be able to use the main results of probability in order to: solve simple problems on probabilities and conditional probabilities; chose probability distributions apt to describe a random phenomenon; appropriately use approximate probability distributions, even for inference.	CONSIGLIATA: Analisi matematica

		1.2 *Capacità di applicare conoscenza e comprensione* Lo studente deve saper usare i principali risultati del calcolo delle probabilità per risolvere problemi su probabilità e probabilità condizionali; saper scegliere le leggi di probabilità idonee alla descrizione di un fenomeno aleatorio; saper utilizzare opportunamente distribuzioni di probabilità approssimate, anche per l'inferenza statistica. **Capacità trasversali / soft skills** **2.1** *Autonomia di giudizio* Lo studente deve essere in grado di modellare opportunamente semplici situazioni di incertezza. **2.2** *Abilità comunicative* Lo studente deve essere in grado di esporre in modo efficace le motivazioni per la scelta della modellazione stocastica di semplici situazioni di incertezza e le conclusioni di una analisi statistica inferenziale. **2.3** *Capacità di apprendimento* Lo studente deve essere in grado di approfondire ulteriori conoscenze nell'ambito della probabilità e dell'inferenza statistica, nell'ambito di analisi di dati da basi di dati, anche in vista dei problemi di sicurezza informatica.	**Cross-sectoral skills/soft skills** **2.1** *Making judgements* The student should be able to model simple situations of uncertainty. **2.2** *Communication skills* The student should be able to discuss the reasons leading to a particular stochastic model in simple uncertainty situations and to explain in an adequate manner the conclusions of a statistical inference. **2.3** *Learning skills* The student should be able to deepen her knowledge in probability and statistical inference, practicing analysis of data from databases, for instance in view of cybersecurity.	
Calcolo scientifico	MAT/08	Il Calcolo Scientifico si occupa dello sviluppo, dell'analisi e dell'implementazione di algoritmi per la risoluzione di problemi della matematica del continuo mediante il calcolatore. L'obiettivo del corso è fornire allo/la studente/essa le conoscenze teoriche di base per risolvere i principali problemi del calcolo scientifico e per analizzare le proprietà di convergenza, stabilità e complessità dei relativi metodi numerici. In	Scientific Computing is concerned with the development, the analysis and the implementation of algorithms for solving mathematical problems involving continuous variables by using the computer. The aim of the course is to present the basic theoretical aspects for solving the main problems of scientific computing and to analyze the convergence, stability and complexity of the	CONSIGLIATE: Analisi Matematica, Matematica discreta

		questo corso, di carattere introduttivo, lo studente/la studentessa conosce - le varie fonti d'errore e le problematiche relative all'elaborazione numerica connesse all'uso del calcolatore (errori di arrotondamento e loro propagazione, procedimenti stabili e instabili, mal condizionamento dei problemi) - i metodi numerici per le equazioni non lineari, la risoluzione di sistemi lineari e l'approssimazione di dati e funzioni e la relativa analisi di convergenza, stabilità e complessità. La risoluzione di esercizi e l'analisi di alcuni casi di studio completano la conoscenza teorica e consentono di imparare ad analizzare criticamente i risultati numerici. Le competenze acquisite permettono di proseguire lo studio della disciplina in ambito più avanzato e forniscono strumenti matematici utili sia per l'informatica che in altri contesti applicativi. Infatti i problemi di calcolo scientifico nascono anche nelle scienze naturali e sociali, nell'ingegneria, nella medicina, nella biologia e nell'economia. Capacità relative alle discipline 1.1 Conoscenza e capacità di comprensione: Lo/la studente/essa deve: conoscere le varie fonti d'errore relative all'uso del calcolatore: la loro origine, la propagazione e alcune tecniche per la loro valutazione; -acquisire la conoscenza teorica per l'analisi dei metodi numerici presentati nel corso; -aver chiaro come la scelta di un algoritmo per la risoluzione di un determinato problema richiede un'analisi del problema (condizionamento) e delle proprietà	related numerical methods. In this introductive course the students learn about - the sources of errors due to the use of the computer (rounding errors and their propagation, well/ill-conditioned problems; stability of algorithms); -the numerical methods for solving nonlinear equations and linear systems, to approximate data and functions, together with the relative analysis of convergence, stability and complexity. The theoretical knowledge is complemented with the solution of exercises and the study of some test examples to learn how critically analyze the results of numerical simulations. The learned skills allow the interested students to continue the study of the discipline at advanced level and, to encompass scientific computing problems arising not only in computer science but also throughout the natural sciences, social sciences, engineering, medicine, and business. Sector-specific skills 1.1 Knowledge and understanding: The course aims to introduce the basic principles of Scientific Computing through the presentation of some mathematical problems, and the study of fundamental algorithms to solve them. The student has to -know the sources of errors due to the use of the computer, understand well/ill-conditioned problems; stability, accuracy and complexity of algorithms;	

		dell'algoritmo (stabilità, accuratezza, costo computazione); -riconoscere, analizzare e risolvere numericamente alcuni problemi della matematica del continuo (equazioni non lineari, sistemi lineari, approssimazione di dati e funzioni) -saper svolgere gli esercizi relativi ad ogni tema trattato nell'insegnamento; -saper stimare l'attendibilità dei risultati numerici ottenuti. 1.2 Conoscenza e capacità di comprensione applicate: Lo/la studente/essa deve acquisire i principi base del Calcolo Scientifico ed essere in grado di applicare il metodo numerico più adatto, attraverso l'esame del problema da risolvere, l'accuratezza richiesta, la complessità e le caratteristiche del calcolatore. Deve essere in grado di valutare le proprietà del metodo numerico e scrivere una pseudocodifica, interpretare i risultati in alcuni casi di studio e trarre le conclusioni. Tali competenze forniscono strumenti matematici utili sia per l'informatica che in altri contesti applicativi, quali le scienze naturali e sociali, nell'ingegneria, nella medicina, nella biologia e nell'economia. Capacità trasversali/soft skills 2.1 Autonomia di giudizio: lo studente acquisisce la capacità di analizzare le proprietà degli algoritmi, di valutare criticamente i risultati delle simulazioni numeriche, e di confrontare tra le prestazioni di diversi algoritmi. Tali abilità mirano	-know how to solve numerically some mathematical problems (nonlinear equations, linear systems, approximation of data and functions) -solve exercises -critically analyze the results of the numerical simulations. 1.2 Applying knowledge and understanding: The student has to know the basic principles of Scientific Computing and to choose the appropriate numerical method, through the analysis of the problem, the required accuracy and the features of the computer. She/he has to be able to evaluate the properties of the numerical method, analyze the numerical results of the case-studies and pull the conclusions. The learned skills allow the students to encompass scientific computing problems arising not only in computer science but also throughout the natural sciences, social sciences, engineering, medicine, and business. Cross-sectoral skills/soft skills 2.1 Making judgments: The student acquires the ability to analyze the properties of algorithms, to critically evaluate the results of numerical simulations and to compare the performance of different algorithms. Such skills aim to develop the maturity of judgment and the critical sense. 2.2 Communication Skills: The student learns to describe correctly and using the appropriate terminology either the	

		a sviluppare la maturità di giudizio e il senso critico. 2.2 Abilità comunicative: lo studente impara a descrivere in modo corretto e usando la terminologia appropriata i problemi matematici e le proprietà dei metodi numerici per la loro risoluzione. 2.3 Capacità di apprendimento: lo studente deve conoscere e analizzare gli algoritmi principali per i problemi base del calcolo scientifico, trarre le conclusioni e comunicare efficacemente.	mathematical problems or the properties of the numerical methods for their resolution. 2.3 Learning skills: The student has to know and to analyze the main algorithms for the basic scientific computing problems, draw conclusions and communicate efficaciously.	
Fisica	FIS/01	Il contenuto disciplinare del corso comprende essenzialmente la Meccanica classica, ed è articolato in: - Introduzione alla fisica sperimentale (misura, grandezze fisiche, modelli fisici) - Cinematica (sistemi di riferimento, grandezze cinematiche, trasformazioni tra sistemi di riferimento) - Dinamica (dinamica del punto materiale, energia/lavoro, dinamica dei sistemi composti) Oltre che fornire una base di conoscenza della Meccanica classica, scopo del corso è quello di fornire agli studenti una panoramica approfondita di cosa sia una scienza sperimentale ed all'interno di essa il ruolo giocato dalla modellizzazione e dal formalismo. Come esiti del corso si attende che gli studenti - Abbiano acquisito conoscenze e capacità di comprensione della fenomenologia e della sua modellizzazione in Meccanica classica	The disciplinary content of the course includes mainly the classical mechanics, and is divided into: -Introduction to experimental physics (measurement, physical quantities, physical models) -kinematics (reference systems, kinematic variables, transformations between coordinate systems) -Dynamics (mass point, energy/work, extended bodies) In addition to a knowledge base of the classical mechanics, the aim of the course is to provide students with a thorough overview of what an experimental science is and, within it, the role played by modelling and formalism. As outcomes of the course students are expected -To have gained knowledge and understanding of the phenomenology and modelling in classical mechanics -To be able to use the modeling tools to address and solve problems	

		- Siano capaci di utilizzare gli strumenti della modellizzazione per affrontare e risolvere problemi - Siano capaci di rielaborare ed esporre in maniera indipendente concetti e metodi appresi nel corso.	-To be able to rework and expose independently the concepts and the methods learned in the course.	
Fondamenti dell'informatica	INF/01	Il corso affronta le basi teoriche della scienza del calcolatore. In particolare, dopo aver superato l'esame si ritiene che lo studente sia in grado di discutere la decidibilità o meno, e la polinomialità o meno, di problemi, anche utilizzando tecniche raffinate di riduzione, e conosca gli aspetti rilevanti della teoria dei linguaggi formali. **Programma** Il corso si divide in tre parti: linguaggi formali, calcolabilità, e complessità. **Linguaggi formali.** Linguaggi regolari. Automi a stati finiti deterministici e non-deterministici, e loro equivalenza. Espressioni regolari. Pumping Lemma per i linguaggi regolari e sue applicazioni. Proprietà di chiusura e di decidibilità dei linguaggi regolari. **Linguaggi liberi dal contesto**. Grammatiche libere dal contesto e alberi di derivazione. Grammatiche ambigue. Le forme normali di Chomsky e di Greibach. Il pumping lemma per i linguaggi liberi e le sue applicazioni. Proprietà di chiusura e di decidibilità dei linguaggi liberi. Grammatiche lineari destre e sinistre, grammatiche di tipo 0 e 1 e gerarchia di Chomsky. **Calcolabilità**. Il concetto di algoritmo. Modelli di calcolo. La macchina di Turing.	We will consider the theoretical foundations of Computer Science. After successfully completing the course the student will be able to discuss the decidability (polinomiality) of relevant problems, using advances reduction techniques, and will know the main notions of formal language theory. **Program** The course is organized in three parts: formal languages, computability, and complexity. **Formal languages.** Regular languages. Deterministic and non-deterministic finite state automata and their equivalence. Regular expressions. The pumping lemma for regular languages and its applications. Closure and decidability properties of regular languages. **Context-free languages**. Context Free Grammars and derivation trees. Ambiguous grammars and languages. Chomsky and Greibach normal forms. The pumping lemma for context-free languages and its applications. Closure and Decidability properties of context free languages. Linear grammars, type 0 and type 1 grammars and Chomsky hierarchy.	VINCOLANTI: Programmazione e laboratorio CONSIGLIATE: Analisi matematica, Matematica discreta, Logica matematica

		Funzioni Turing-calcolabili. Enumerazione. Halting Problem e sua indecidibilità. Il teorema SMN. I formalismi delle funzioni primitive ricorsive e ricorsive parziali. La funzione di Ackermann. Equivalenza tra le funzioni ricorsive parziali e le funzioni Turing-calcolabili. Teoremi di Rice, Rice-Shapiro e i Teoremi di ricorsione. Applicazioni alla programmazione. Riducibilità funzionale. Studio della relazione. Insiemi ricorsivi e completi. Insiemi creativi e produttivi. Teorema di Myhill. Insiemi semplici: esistenza di un insieme semplice. **Complessità.** Problemi e linguaggi. Problemi decisionali e funzionali. Classi di complessità in tempo deterministiche e non deterministiche; classi in spazio: principali risultati. Riduzioni, completezza, ed esempi. Teoremi di Cook. NP-completezza di alcuni problemi fondamentali mediante riduzione. **Capacità relative alle discipline** **1.1** Conoscenza *e capacità di comprensione*: lo studente acquisisce conoscenze logico-formali fondanti la disciplina ``informatica''. In particolare, le nozioni di linguaggio formale e la classificazione degli stessi secondo Chomsky, la nozione di funzione calcolabile e le sue principali caratterizzazioni e limiti, la nozione di complessità di un problema e le principali tecniche per studiarla. **1.2** *Conoscenza e capacità di comprensione applicate*: lo studente sarà in grado di comprendere la complessità di un linguaggio	**Computability.** Models for computation. Turing Machine. Turing-computability, enumeration, Halting problem and Theorem SMN. (Primitive) recursive functions. Equivalence theorem between the two formalisms. Kleene recursion theorems, Rice and Rice-Shapiro theorems and their applications. Reducibility between sets. Recursive and complete sets. The Myhill Theorem. Simple Sets: definition, properties and their existence. **Complexity.** Languages and Problems. Deterministic and non-deterministic time complexity classes; space classes: main results and relative relationships. Reductions and Completeness. Cook's Theorems. NP-completeness of some fundamental problems using reductions. **Sector-specific skills** **1.1** Knowledge *and understanding*: the student will acquire the basic theoretical notions at the basis of computer Science. In particular, she/he will know and understand the notions of formal language and its classification according to Chomsky, the notion of computable function and its main characterizations and limits, the notion of complexity of a problem and the main techniques for studying it. **1.2** *Applying knowledge and understanding*: being capable of understanding the complexity of a formal language, the student	

		formale e dunque quale sia lo strumento software più adatto per generarlo o riconoscerlo, di comprendere se un problema sia o meno risolubile con un calcolatore e, nel primo caso, di comprendere se sia possibile risolverlo efficientemente. **Capacità trasversali/soft skills** **2.1** *Autonomia di giudizio:* lo studente acquisisce una capacità di valutazione critica sulle specifiche di un problema che permettono di comprenderne il suo grado di calcolabilità e, nel caso, sia calcolabile, il suo grado di complessità. **2.2** *Abilità comunicative*: lo studente impara ad utilizzare la terminologia precisa nell'area dei linguaggi formali (automi, grammatiche), della calcolabilità (problema decidibile, semi decidibile etc) e della complessità (problema polinomiale, classe NP, etc). **2.3** *Capacità di apprendimento*: lo studente impara a familiarizzare con il metodo scientifico applicato in modo rigoroso alla disciplina informatica. Sebbene il testo principale sia in italiano, gran parte dei riferimenti suggeriti sono in lingua inglese e pertanto lo studente migliora la sua capacità di comprendere informazioni ricevute in tale lingua.	will be able to choose the best software tool for generating or recognizing it. The student will be capable of understanding whether a problem can be solved by a computer or not, and, in the first case to understand if it can be solved efficiently or not. **Cross-sectoral skills/soft skills** **2.1** *Making judgements:* given the formal specifics of a problem, the student will acquire the capability of deciding whether it can be solved or not by a computer and, in the first case, its degree of complexity. **2.2** *Communication skills*: the student will learn the exact terminology of the foundations of computer science and therefore to use properly: notions of formal languages theory (automata, grammars, etc), of computability (decidable problem, semi decidable problem, productive, creative, simple sets, etc), and of complexity (polynomial problems, NP class, etc). **2.3** *Learning skills*: the student will learn how the scientific method can be applied to computer science. Moreover, being the auxiliary suggested references in English, she/he will improve her/his language skills.	
Ingegneria del software	ING-INF/05	Gli obiettivi del corso sono i seguenti: - Introdurre ai concetti di base dell'Ingegneria del Software (IS), ossia il settore ingegneristico dell'informatica dedicato allo studio delle metodologie, delle tecniche e degli strumenti utilizzati in tutti gli aspetti della produzione industriale del software.	The course objectives are: - Introducing basic concepts, principles, and techniques of Software Engineering (SE), i.e. the discipline of Computer Science devoted to the professional and industrial development of software.	VINCOLANTE: Programmazione e laboratorio CONSIGLIATA: Programmazione orientata agli oggetti

		- Capire quali sono i principali problemi che si incontrano nello sviluppo del software a livello professionale ed industriale. - Conoscere come è organizzato il processo di produzione e di sviluppo del software **Capacità relative alle discipline** Lo studente dovrà: 1. *Conoscenza e capacità di comprensione*: acquisire specifiche conoscenze dei principali concetti e principi dell'IS. Conoscere molteplici modelli del processo di sviluppo del software e le varie fasi del ciclo di vita, con riferimento sia al software tradizionale, che al software avanzato. Conoscere diverse tecniche di analisi e di progettazione del software, ivi inclusi i linguaggi: UML, Data Flow Diagram e Reti di Petri. Conoscere i maggiori fattori di criticità nello sviluppo del software. Conoscere le principali tecniche di verifica e validazione del software, ivi incluse le tecniche di testing e di software inspection. Conoscere i principali parametri di qualità ed i principali standard del software. Conoscere le nuove tecniche agili di sviluppo del software. 2. *Capacità di applicare conoscenza e comprensione:* sapere come analizzare e rappresentare i risultati dell'analisi e della progettazione in uno specifico dominio applicativo. Saper selezionare le metodologie e le tecniche di sviluppo più adeguate alle varie situazioni. **Capacità trasversali / soft skills**	- Understanding the main problems arising during the software development process. - Knowing how the software development process is organized **Sector-specific skills** The student will have to: 1. *Knowledge and understanding:* acquiring specific knowledge of the main concepts and basic principles of SE. Knowing the various development models exploited for producing software, the various phases of the software life cycle, both for traditional software and for innovative software. Knowing the different techniques for analysis and design, including languages such as UML, Data Flow Diagrams, and Petri Nets. Knowing the major critical problems encountered in software development. Knowing the main verification and validation techniques, including testing and software inspection. Knowing the main quality criteria and standards for software. Knowing the new agile methodologies for software development. 2. *Applying knowledge and understanding*: knowing how to analyze and represent the results of analysis and design in a specific application context. Knowing how to select the most appropriate technique for the specific situation considered. **Cross-sectoral skills/soft skills** The student will have to:	

		Lo studente dovrà: 1. *Autonomia di giudizio*: saper valutare indipendentemente le caratteristiche ed i requisiti di un'applicazione informatica ed essere in grado di rappresentarne l'analisi e la progettazione. 2. *Abilità comunicative:* saper illustrare con rigore logico e terminologico, a voce e per iscritto mediante varie tecniche di rappresentazione, gli elementi tecnici rilevanti di un progetto software. 3. *Capacità di apprendimento*: essere in grado di apprendere le nozioni base dell'IS, al fine poi all'occorrenza di raffinare e di approfondire specifici settori della disciplina.	1. *Making judgments:* be able to independently evaluate the characteristics of a computer application and to be able to represent the results of analysis and design. 2. *Communication skills*: acquiring the ability to describe effectively and through appropriate models the most relevant aspects of a software project 3. *Learning skills*: be able to learn the basics of SE, in order to possibly later refine and deepen specific areas of the discipline.	
Interazione uomo-macchina	INF/01	Scopo del corso è di introdurre i principi, le metodologie di progettazione e le diverse scelte implementative per la costruzione di software che sia usabile in modo semplice, intuitivo, produttivo, affidabile e piacevole dagli utenti a cui è rivolto. Il raggiungimento di questo obbiettivo richiede lo studio di tre diverse tematiche: i fattori umani (caratteristiche psicologiche dell'utente); la macchina (possibilità offerte dalle periferiche di input/output esistenti); l'interazione (analisi, progetto, valutazione di interfacce utente). Oltre a presentare le nozioni di base della disciplina, il corso pone anche l'accento su alcuni sviluppi recenti di particolare importanza, quali il groupware, le interfacce 3D, i social network e la comunicazione fra persone in rete. Il corso prevede anche la partecipazione ad una reale valutazione su utenti di un'interfaccia in modo da consentire allo studente di veder applicate	The aim of the course is to introduce the principles, design methodologies and the various implementation choices to build software that can be used in a simple, intuitive, productive, reliable and enjoyable way by the users to whom it is addressed. The achievement of this goal requires the study of three different themes: human factors (psychological characteristics of the user); the machine (possibility offered by existing input / output devices); interaction (analysis, design, evaluation of user interfaces). In addition to presenting the core concepts of the discipline, the course also emphasizes some recent major developments such as groupware, 3D interfaces, social networks, and on-line communication between people. The course also includes participation in a real evaluation of a user interface so that the student can	VINCOLANTI: Programmazione e laboratorio, Architettura degli elaboratori e laboratorio

		concretamente le tecniche di valutazione apprese nelle lezioni. **Capacità relative alle discipline** **1.1** Conoscenza *e capacità di comprensione*: gli studenti acquisiscono conoscenze specifiche multidisciplinari sulle caratteristiche degli utenti e delle interfacce ad essi rivolte. Essi inoltre imparano a valutare ed a scegliere fra varie tecniche di interazione, a seconda degli obbiettivi del sistema informatico a cui è destinata l'interfaccia, del suo contesto d'uso e del suo utente target. **1.2** *Conoscenza e capacità di comprensione applicate*: grazie ad una serie di casi di studio reali presentati a lezione ed alla partecipazione ad una reale attività di valutazione interfaccia su utenti, lo studente acquisisce specifiche capacità di applicare a casi reali le conoscenze maturate sui vari aspetti della disciplina. **Capacità trasversali/soft skills** 2.1 *Autonomia di giudizio:* lo studente acquisisce una capacità di valutazione critica delle diverse caratteristiche di un'interfaccia utente e di come esse possono influire positivamente o negativamente sull'efficacia dell'interfaccia in diversi contesti d'uso e per diverse categorie di utenza. 2.2 *Abilità comunicative*: lo studente impara a descrivere un'interfaccia utente in modo tecnicamente corretto ed usando la terminologia appropriata. Il corso inoltre dedica alcune lezioni al tema della comunicazione interpersonale di tipo mediato attraverso strumenti informatici quali e-mail, videoconferenza e social network.	effectively apply the evaluation techniques learned in the course. **Sector-specific skills** 1.1 *Knowledge and understanding*: Students acquire specific multidisciplinary knowledge on the characteristics of users and human-computer interfaces. They also learn to evaluate and choose from various interaction techniques, depending on the objectives of the computer application for which the interface is intended, its context of use, and its target user. 1.2 *Applied knowledge and understanding*: Through a series of case studies and participation in a real evaluation of a user interface, students acquire specific skills to apply knowledge of the discipline to the various aspects of real-world projects concerning user interfaces. **Cross-sectoral skills/soft skills** 2.1. *Making judgments*: Students acquire the ability to critically evaluate the different features of a user interface and how they can positively or negatively affect the effectiveness of the interface in different contexts of use and for different categories of users. 2.2 *Communication Skills*: Students learn to describe a user interface in a technically correct way and using the appropriate terminology. The course also devotes some lessons to the topic of interpersonal communication mediated through computer tools such as e-mail, videoconferencing and social networks.	

		2.3 *Capacità di apprendimento*: il corso fornisce le basi e gli strumenti che permettono allo studente di approfondire ed affrontare autonomamente problemi inerenti alla progettazione e valutazione di interfacce utente.	2.3 *Learning Skills*: The course provides the knowledge and tools that enable the student to deepen and address autonomously issues related to the design and evaluation of user interfaces.	
Linguaggi di programmazione	INF/01	Nei primi anni del corso di laurea vengono presentati diversi linguaggi di programmazione: Assembly, C, Scheme, Java che usano diversi paradigmi: imperativo, funzionale, ad oggetti. Scopo del corso è approfondire queste conoscenze concentrandosi sui principi "universali" che guidano la progettazione, realizzazione e implementazione dei moderni linguaggi di programmazione. Viene data una panoramica complessiva dei vari paradigmi di programmazione, mostrando quali siano vantaggi e svantaggi di ciascun paradigma, e come questi vadano sfruttati nello scrivere codice. Per dare concretezza al corso e illustrare meglio i concetti presentati vengono introdotti ulteriori linguaggi di programmazione quali Haskell e Erlang. Vengono inoltre presentati argomenti relativi alla descrizione del comportamento dei programmi mediante macchine astratte, e all'implementazione dei linguaggi attraverso i compilatori. **Capacità relative alla disciplina** *1.1 Conoscenza e capacità di comprensione*: dopo aver superato l'esame si ritiene che lo studente conosca i principali concetti e paradigmi alla base dei linguaggi di programmazione. 1.2 *Conoscenza e capacità di comprensione applicate*: lo studente impara nuovi linguaggi di	In the early years of the degree course, various programming languages are presented: Assembly, C, Scheme, Java, which use different paradigms: imperative, functional, object-oriented. The aim of the course is to deepen this knowledge by focusing on the "universal" principles that guide the design and implementation of modern programming languages. An overview of the various programming paradigms is given, showing what are the advantages and disadvantages of each paradigm. To better illustrate the above concepts, further programming languages are introduced, such as Haskell and Erlang. Further arguments are the description of programs behavior by abstract machines, and the implementation of languages through compilers. **Sector-specific skills** *1.1 Knowledge and understanding*: After passing the exam, the student knows the main concepts and paradigms of programming languages. 1.2 *Applying knowledge and understanding*: Students learn to write code in new	CONSIGLIATE: Algoritmi e strutture dati e laboratorio, Fondamenti dell'informatica, Programmazione e laboratorio, Programmazione orientata agli oggetti, Sistemi operativi e laboratorio

		programmazione e a scrivere codice nei vari paradigmi usando lo stile e le tecniche appropriate. **Capacità trasversali/soft skills** 2.1 *Autonomia di giudizio*: obiettivo del corso è anche quello di sviluppare uno spirito critico che permetta di saper scegliere il paradigma più adatto a seconda del problema applicativo da risolvere, sapendo quali costrutti di un linguaggio usare, e a quale costo. 2.2 *Abilità comunicative*: lo studente impara l'esatto significato dei termini usati nella programmazione. 2.3 *Capacità di apprendimento*: le conoscenze apprese permettono allo studente muoversi con una certa disinvoltura tra la miriade di linguaggi di programmazione riuscendo a inquadrare velocemente le caratteristiche di nuovo linguaggio, e a impararlo in breve tempo.	programming languages using the appropriate style. **Cross-sectoral skills / soft skills** 2.1. *Making judgment*: one of the course objective is also to develop a critical spirit that allows to choose the most suitable paradigm depending on the problem to be solved. 2.2 *Communication skills*: Students learn the exact meaning of terms used in programming. 2.3 *Learning skills*: The knowledge learned allows the student to move with some ease among the plethora of programming languages, by learning them in a short time.	
Logica matematica	MAT/01	Obiettivi Formativi Specifici. • Acquisire gli elementi di base della logica matematica, con particolare attenzione ai metodi algoritmici. • Raggiungere la capacità di tradurre affermazioni dal linguaggio naturale al linguaggio formale. • Sviluppare una buona conoscenza degli aspetti sintattici e semantici della logica proposizionale e predicativa. Lo/la studente/essa dovrà: **Capacità relative alle discipline** 1.1. *Conoscenza e capacità di comprensione* Conoscere i concetti base della logica matematica: sintassi e semantica della logica proposizionale e predicativa, validità,	• To acquire the fundamentals of mathematical logic, paying special attention to algorithmic methods. • To reach the ability of translating sentences of a natural language into a formal language. • To reach a good knowledge of the syntactic and semantic aspects of both propositional and predicative logic. The student will have to: **Sector-specific skills** *1.1. Knowledge and understanding* To know the basic notions of mathematical logic: syntax and semantics of propositional and predicative logic, validity, satisfiability, logical consequence and equivalence,	CONSIGLIATA: Matematica discreta

		soddisfacibilità, conseguenza ed equivalenza logica, elementare equivalenza. Conoscere le nozioni di correttezza e completezza di un calcolo logico e, in alcuni casi, saperle dimostrare. 1.2 *Capacità di applicare conoscenza e comprensione* Saper trasformare algoritmicamente una formula in forma normale o in forma prenessa. Saper tradurre asserzioni dal linguaggio naturale al linguaggio formale. Saper riconoscere validità, soddisfacibilità e conseguenza logica utilizzando calcoli logici quali le tavole di verità, i tableaux e la deduzione naturale. Saper costruire semplici deduzioni naturali che modellano ragionamenti corretti. **Capacità trasversali / soft skills** 2.1 *Autonomia di giudizio* Conoscere pregi e limiti delle varie formalizzazioni logiche. Saper scegliere quale linguaggio, calcolo logico o algoritmo utilizzare per formalizzare un'asserzione formulata nel linguaggio naturale in modo da poterne verificare la correttezza. 2.2 *Abilità comunicative* Essere in grado di motivare, a voce e per iscritto, le proprie scelte relative al punto precedente. Saper spiegare perché un'asserzione non è logicamente corretta. 2.3 *Capacità di apprendimento* Individuare le eventuali espansioni delle logiche studiate necessarie per la formalizzazione e lo studio di asserzioni più complesse.	elementary equivalence. To know the notions of soundness and completeness of a logical calculus and, in some cases, to be able to prove these properties. *1.2 Applying knowledge and understanding* To be able to algorithmically transform a formula in normal form or in prenex form. To be able to translate assertions and arguments from a natural language into a formal language. To be able to identify validity, satisfiability and logical consequence using logical calculi such as truth tables, tableaux and natural deduction. To be able to construct simple natural deductions which model correct arguments. **Cross-sectoral skills/soft skills** *2.1 Making judgements* Know strength and limits of various logical formalizations, To be able to choose the language, logical calculus or algorithm most appropriate for the formalization a statement in the natural language, with the goal of verifying its correctness. *2.2 Communication skills* To be able to explain, verbally and in writing, his/her choices relative to the previous item. To be able to explain why an argument is not logically sound. *2.3 Learning skills* To identify the appropriate expansions of the logic, which are needed to formalize and study more complex statements.	
Matematica discreta	MAT/09	Obiettivi formativi specifici Gli obiettivi culturali del corso sono: acquisire i concetti e gli strumenti elementari di aritmetica e	Specific educational objectives The goals of the course are to provide the basic tools and concepts of arithmetic,	

		calcolo combinatorio utili all'informatica e fornire i concetti fondamentali dell'algebra lineare. Dopo aver superato l'esame, si ritiene che lo studente abbia familiarità con i concetti di: 1. Insiemi, funzioni e relazioni. 2. Numeri naturali ed interi, divisibilità, congruenze, induzione. Scomposizione in fattori primi con applicazioni 3. Sommatorie e loro manipolazioni. 4. Elementi di calcolo combinatorio e numeri di Fibonacci. 5. Elementi di teoria dei grafi quali: Connessione, cammini e cicli, alberi, circuiti hamiltoniani ed euleriani, accoppiamenti, cliques e insiemi indipendenti, grafi bipartiti. 6. Spazi vettoriali, dipendenza ed indipendenza lineare, basi e dimensione. Combinazioni lineari convesse, insiemi convessi. 7. Spazio Euclideo delle n-ple. 8. Matrici e calcolo matriciale. 9. Applicazioni lineari. Relazione tra applicazioni lineari e matrici. Inversa e determinante. 10. Sistemi Lineari. Autovalori ed autovettori di un'applicazione lineare. Diagonalizzazione. Capacità relative alle discipline: 1.1 Conoscenza e comprensione: lo studente acquisisce la conoscenza del linguaggio matematico di base e dei principali strumenti del ragionamento matematico formale e rigoroso. Impara alcune tecniche fondamentali del theorem proving, quali la dimostrazione per assurdo e l'induzione. Impara i fondamenti dell'algebra lineare e degli spazi vettoriali, con particolare riguardo allo spazio euclideo n-	combinatorics and linear algebra for being used in computer science. After completion of the course, the student has acquired a familiarity with the following concepts: 1. Sets, functions, relations. 2. Integer and natural numbers, divisibility, congruence, induction. Prime factorization with its applications. 3. Sums and multiple sums. 4. Combinatorics and Fibonacci's numbers. 5. Graph theory: connectivity, paths, cycles, trees, hamiltonian and eulerian circuits, matchings, cliques, stable sets, bipartite graphs. 6. Vector spaces, linear independence, basis, dimension. Convex combinations, convex sets. 7. The euclidean space of n-tuples. 8. Matrices and matrix computations. 9. Linear functions and their relations with matrices. Inverse and determinant. 10. Linear systems. Eigenvalues and eigenvectors. Diagonalization. Sector-specific skills 1.1 Knowledge and understanding: the student acquires the knowledge of the basic mathematical language and of the main tools for mathematical reasoning. He learns some fundamental techniques of problem solving, such as the use of induction and contradiction. He learns the fundamentals of linear algebra and vector spaces, in particular the euclidean n-dimensional space.	

		dimensionale. 1.2 Capacità di applicare conoscenza e comprensione: Lo studente sa manipolare sommatorie e calcoli complessi sugli interi. È in grado di risolvere sistemi lineari di medio/piccole dimensioni, sa calcolare l'inversa di piccole matrici, e il loro determinante. Sa risolvere problemi di natura geometrica e interpretare geometricamente le applicazioni lineari. Capacità trasversali /soft skills 2.1 Autonomia di giudizio: Lo studente è in grado di formulare ipotesi alternative sulla soluzione di un problema di natura discreta, conosce la crescita delle funzioni esponenziali e capisce per quali istanze di problema sia possibile un'enumerazione delle soluzioni e per quali no. Conosce le principali tecniche di dimostrazione (quali l'assurdo, e l'induzione) ed è in grado di dimostrare semplici teoremi. 2.2 Abilità comunicative: Lo studente è fluente nel linguaggio fondamentale della matematica discreta/algebra lineare ed è cosciente dell'importanza della notazione rigorosa che in esse viene applicata. 2.3 Capacità di apprendimento: Lo studente apprende gli strumenti fondamentali della matematica, dalla logica alla teoria degli insiemi, alle sommatorie che costituiranno la base a molti dei suoi futuri corsi di studio.	1.2 Applying knowledge and understanding: The student can manipulate complex expressions and sums over the integers. He can solve small-to-medium sized systems of linear equations. He can compute the inverse of small matrices and their determinant. He can solve some geometric problems and give a geometrical interpretation to linear maps and vector spaces. Cross-sectoral skills/soft skills 2.1 Making judgements: The student can make alternative hypotheses about the solution of a discrete problem, he is familiar with an exponential growth rate and he understands when a problem instance can be solved by enumerating its solutions and when it cannot. He knows the main proving techniques (such as induction and contradiction) and he can prove some simple theorems. 2.2 Communication skills: The student is fluent in the basic language of discrete mathematics and linear algebra. He understands the importance of the rigorous notation that they must employ. 2.3 Learning skills: The student acquires the basic tools of modern mathematics, from boolean algebra, to sets and functions, sums, etc, which will be instrumental to many of his future courses.	
Programmazione e Laboratorio	INF/01	Il corso introduce i fondamenti scientifici e metodologici della programmazione e si propone	The course introduces the scientific and methodological foundations of computer	

Foundations of Computer Programming		di mettere lo studente in condizione di cogliere pienamente la natura di questa attività, stimolando inoltre un atteggiamento critico nei confronti degli strumenti utilizzati. L'approccio seguito è di tipo functional-first (IEEE-CS/ACM Computing Curricula 2001); il percorso didattico si articola attorno ai principali processi di astrazione che consentono di gestire la complessità dei problemi affrontati in ambito informatico: l'astrazione procedurale, l'astrazione sui dati e l'astrazione sullo stato. Al termine del corso lo studente dovrebbe aver acquisito le competenze di base e le capacità operative necessarie al fine di progettare, organizzare e formalizzare programmi di piccole dimensioni, sviluppati secondo i paradigmi funzionale, imperativo e object-oriented; dovrebbe inoltre essere in grado di analizzare la struttura logica di un programma al fine di verificarne la correttezza in relazione alle specifiche. Le attività di Laboratorio consistono nel progetto, nello sviluppo e nella sperimentazione di programmi di piccole dimensioni, e hanno lo scopo di stimolare le capacità organizzazione e lavoro autonomo da parte degli studenti. **Programma Sintetico** I. Astrazione procedurale: - Procedure come astrazione di espressioni; - Ricorsione, ricorsione ad albero e ricorsione di coda; - Argomenti e valori procedurali; - Correttezza di programmi ricorsivi. II. Astrazione sui dati: - Tipi semplici e tipi strutturati;	programming, in order for students to understand the nature of programming and to develop a critical attitude toward the related tools. The approach is functional-first (IEEE-CS/ACM Computing Curricula 2001); the course is organized around the main forms of abstraction, which are exploited to deal with problem complexity: procedural abstraction, data abstraction, and abstraction of state. At the end of the course, the student should have acquired the basic knowledge and skills to design organize and write small-scale programs, developed according to the functional, imperative and object-oriented paradigms; moreover, he/she should be able to analyze the logic of a program to verify its correctness with respect to its formal specifications. The laboratory sessions are about design, development and experimentation of small-scale programs; they are meant to stimulate students' organization skills as well as their ability to work autonomously. **Short Syllabus** I. Procedural abstraction: - Expressions and procedures; - Recursion, tree recursion and tail recursion; - Higher order functions; - Correctness of recursive programs. II. Data abstraction: - Simple and structured types; - Operations on abstract data; - Use vs. implementation of abstract data. III. Abstraction of state:	

		- Protocollo relativo a un dato astratto; - Utilizzo e realizzazione di un dato astratto. III. Astrazione sullo stato: - Concetto di stato e paradigma imperativo; - Tecniche di memoization (top-down) e programmazione dinamica (bottom-up); - Classi, oggetti e incapsulamento dello stato; - Programmazione orientata agli oggetti; - Correttezza di programmi iterativi. **Capacità relative alle discipline** Lo/la studente/essa dovrà: 1.1. *Conoscenza e capacità di comprensione* - Acquisire le basi della programmazione funzionale e imperativa; - Acquisire i primi rudimenti della programmazione object-oriented; - Conoscere sintassi e semantica dei principali costrutti e delle principali strutture dati nei linguaggi Scheme e Java; - Conoscere le relazioni fra ricorsione e iterazione; - Conoscere le relazioni fra i concetti di classe e oggetto. 1.2 *Capacità di applicare conoscenza e comprensione* - Saper codificare semplici algoritmi ricorsivi e imperativi; - Saper analizzare la logica di un semplice programma ricorsivo o imperativo per verificarne la correttezza rispetto alle specifiche e per stimarne le prestazioni; - Saper riconoscere quando è appropriato e saper applicare tecniche di memoization e programmazione dinamica per migliorare le prestazioni di programmi ricorsivi;	- Concept of state and imperative paradigm; - Memoization (top-down) and dynamic programming (bottom-up); - Classes, objects and information-hiding; - Object-oriented programming; - Correctness of iterative programs. **Sector-specific skills** 1.1. *Knowledge and understanding* - Basic knowledge of functional and imperative programming; - First rudiments of object-oriented programming; - Basic knowledge of the syntax and semantics of the main program and data structures in Scheme and Java; - Knowledge of the relationships between recursion and iteration; - Knowledge of the relationships between the concepts of class and object. 1.2 *Applying knowledge and understanding* - Being able to code simple recursive and imperative algorithms; - Being able to analyze the logic of a simple recursive or imperative program in order to verify its correctness with respect to the given specifications and to estimate its performances; - Being able to apply the memoization and dynamic programming techniques to improve the performances of recursive programs; - Being able to implement a class with a given interface. **Cross-sectoral skills/soft skills** 2.1 *Making judgements*	

		- Saper realizzare una classe rispettando il protocollo specificato. **Capacità trasversali / soft skills** Lo/la studente/essa dovrà: 2.1 *Autonomia di giudizio* - Saper analizzare i problemi al fine di identificare gli aspetti che si prestano ad essere affrontati attraverso programmi, esprimere le specifiche in maniera precisa e scegliere gli strumenti di programmazione adatti allo scopo; - Saper applicare le proprie conoscenza per capire i processi di elaborazione dell'informazione. 2.2 *Abilità comunicative*. - Essere in grado di confrontarsi con i pari al fine di progettare o migliorare la soluzione algoritmica di un problema. 2.3 *Capacità di apprendimento* - Essere in grado di attuare sperimentazioni sistematiche di un programma per verificarne la correttezza e per stimarne le prestazioni; - Essere in grado di orientarsi nell'ambito dei linguaggi di programmazione e di apprendere autonomamente nuovi linguaggi.	- Being able to analyze problems in order to identify what can be achieved by a program, to formalize appropriate specifications and to choose appropriate programming tools; - Being able to understand information-processing processes. 2.2 *Communication skills*. - Being able to work together with peers in order to design or improve the algorithmic solution of a problem. 2.3 *Learning skills* - Being able to plan experiments on a program to test its correctness and to estimate its performances; - Being able to learn autonomously new programming languages	
Programmazione orientata agli oggetti	INF/01	Scopo del corso è la comprensione profonda dei principi della programmazione orientata agli oggetti e di come questi influenzano la qualità del codice e la sua manutenibilità. Argomenti Il corso prevede di formare lo studente relativamente a questi argomenti: La programmazione in the large vs in the small: il ruolo della qualità nello sviluppo del software. Tipi di astrazione: procedurale, parametrica, per specifica. L'approccio “clean code”. Programmi come funzioni totali o parziali; le eccezioni; la	The aim of the course is to ground the principles of object orientation and how they are related to quality of the code and its maintenability. Topics The following topics are covered: programming in the large. Abstractions in procedures, parameterization, by specification.	VINCOLANTE: Programmazione e laboratorio

		programmazione difensiva e quella assertiva. I tipi di dati astratti. Gerarchie di tipi; le interfacce; le classi astratte; il principio dell'inversione della dipendenza. Il principio di Liskov. Tipi generici. Classi anonime. Cenni alla progettazione orientata agli oggetti: la tecnica CRC, il principio della singola responsabilità, alcuni pattern di design (factory, builder, observer, decorator). Capacità relative alla disciplina Conoscenza e capacità di comprensione conoscere i concetti di base della orientazione agli oggetti conoscere alcuni pattern di design orientato agli oggetti conoscere i 7 principi SOLID di progettazione (singola responsabilità, Liskov, inversione dipendenza, open-closed, interface segregation) e di programmazione (clean code) Capacità di applicare conoscenza e comprensione saper utilizzare Java per la programmazione orientata agli oggetti saper applicare alcuni pattern di design. Questi obiettivi saranno supportati da attività guidata di laboratorio. Capacità trasversali Autonomia di giudizio saper valutare la qualità del codice prodotto Abilità comunicative essere in grado di descrivere e motivare le scelte fatte Capacità di apprendimento essere in grado di approfondire autonomamente gli argomenti toccati a lezione	The “clean code” approach. Defensive vs assertive programming and the role of exceptions. Type hierarchies, Liskov principle, dependency inversion principle. Generics, anonymous classes. OO design, CRC techinque, design patterns (factory, builder, observer, decorator). Sector specific topics Knowledge and understanding Basic concepts of object oriented programming; basic design patterns; SOLID design principles (single responsibility, Liskov, dependency inversion, open-closed, interface segregation); the clean code approach. Applying knowledge and understanding Being able to use Java and common IDEs; being able to apply design patterns. These objectives will be supported by laboratory work. Cross sectoral skills Making judgements Being able to assess the quality of the code being produced. Communication skills being able to describe technical choices and to argue the validity of design decisions.	

			Learning skills Being able to deepen one's own understanding of the topics covered during lectures.	
Reti di calcolatori	INF/01	Scopo del corso è di introdurre i principi fondamentali delle moderne reti di calcolatori in modo da fornire allo studente le conoscenze necessarie per l'analisi e la progettazione di una rete di calcolatori. In particolare verranno presentate le caratteristiche generali delle reti, dei segnali a livello fisico, i modelli ISO/OSI e TCP/IP, e le problematiche affrontate dai principali protocolli ad ogni livello di questi modelli. Inoltre vengono affrontate le problematiche di sicurezza delle comunicazioni tra calcolatori; a questo scopo, verranno introdotte anche alcune nozioni di crittografia. **Capacità relative alle discipline** 1.1 *Conoscenza e capacità di comprensione:* lo studente acquisisce le conoscenze e capacità necessarie per analizzare e progettare una rete di calcolatori, e comprendere eventuali problematiche ad esse connesse. Le conoscenze coprono i molteplici aspetti delle reti: architettura di rete, modello client/server, programmazione socket, accesso alla rete condivisa cablata e wireless, principi di internetworking, protocolli di trasporto, controllo della congestione, codifica dei dati multimediali, aspetti di sicurezza e crittografia. 1.2 *Conoscenza e capacità di comprensione applicate:* grazie ad una serie di esempi presentati a lezione e esercizi mirati alla risoluzione di casi pratici, lo	The aim of the course is to introduce the basic principles of modern computer networks in order to provide the student with the knowledge needed for the analysis and design of a computer network. Specifically, the course presents the general characteristics of networks, physical-level signals, ISO/OSI and TCP/IP models, and the issues addressed by the major protocols at each level of these models. In addition, the security issues of computer communications are addressed; to this purpose, some notions of cryptography will also be introduced. **Skills related to the disciplines** 1.1 *Knowledge and understanding*: The student acquires the knowledge and skills needed to analyze and design a computer network, and to understand related issues. This knowledge covers the many aspects of networks: network architecture, client/server model, socket programming, wired and wireless shared network access, internetworking principles, transport protocols, congestion control, encoding of multimedia data, security aspects and encryption. 1.2 *Applied knowledge and understanding*: By means of a series of examples presented in class and exercises aimed at solving practical cases, the student will learn how to apply to	VINCOLANTE: Sistemi operativi e laboratorio CONSIGLIATA: Programmazione e laboratorio

		studente acquisisce specifiche capacità di applicare a casi reali le conoscenze maturate sui vari aspetti della disciplina. **Capacità trasversali/soft skills** 2.1 *Autonomia di giudizio:* lo studente acquisisce una capacità di valutazione critica delle diverse caratteristiche di una rete di calcolatori e di come esse possono influire sul funzionamento di un sistema informativo aziendale o di una applicazione distribuita, in diversi contesti d'uso e per diverse categorie di utenza. 2.2 *Abilità comunicative*: lo studente impara a descrivere e progettare una rete di calcolatori in modo tecnicamente corretto ed usando la terminologia appropriata. Questo permette allo studente di inserirsi rapidamente in contesti professionali in cui questa terminologia tecnica è essenziale. 2.3 *Capacità di apprendimento*: il corso fornisce le basi e gli strumenti che permettono allo studente di approfondire ed affrontare autonomamente problemi inerenti all'analisi, progettazione e realizzazione di una rete di calcolatori.	real-life cases the knowledge about the various aspects of the discipline. **Cross skills / soft skills** 2.1. *Autonomy of judgment*: the student acquires the capability of assessing the different characteristics of a computer network and how they can affect the functioning of a business information system or distributed application in different contexts and for different categories of users. 2.2 *Communicative Skills*: the student learns how to describe and design a computer network in a technically correct manner and using the appropriate terminology. This allows the student to be acquainted in professional contexts where this technical terminology is essential. 2.3 *Learning Skills*: The course provides the basics and tools that enable the student to deepen and face independently issues related to the analysis, design and realization of a computer network.	
Sistemi operativi e laboratorio	INF/01	**Capacità relative alla disciplina** *Conoscenza e capacità di comprensione*: lo studente acquisisce conoscenze specifiche sull'architettura, il funzionamento e sui principi di progettazione di un moderno sistema operativo. Egli inoltre impara a valutare ed a scegliere l'adozione di un sistema operativo a seconda dell'ambito di utilizzo e delle caratteristiche dell'hardware. *Capacità di applicare conoscenza e comprensione*: grazie all'attività di laboratorio, lo studente	**Sector-specific skills** *Knowledge and understanding*: the student will acquire specific knowledge about the architecture, the operation and the underlying design principles of a modern operating system. She/He will also learn how to evaluate and how to choose among several operating systems, depending on the purpose of the implementation and on the hardware features.	CONSIGLIATE: Architettura degli elaboratori e laboratorio, Programmazione e laboratorio

		acquisisce specifiche capacità di applicare a casi reali le conoscenze maturate sui sistemi operativi. **Capacità trasversali/soft skills** *Autonomia di giudizio*: lo studente acquisisce una capacità di valutazione critica sulle tecnologie, gli algoritmi e le soluzioni che possono influire positivamente o negativamente sulle prestazioni e sul corretto funzionamento di un sistema operativo in diversi contesti applicativi. Abilità comunicative: lo studente impara a descrivere in modo tecnicamente corretto ed usando la terminologia appropriata un sistema operativo. *Capacità di apprendimento*: lo studente impara ad essere autonomo nell'espandere le proprie conoscenze sui sistemi operativi oltre le nozioni e gli esempi appresi ed analizzati durante il corso, scoprendo e sfruttando anche le connessioni con altri corsi riguardanti l'architettura dei calcolatori, le reti ed i sistemi distribuiti.	*Applying knowledge and understanding*: thanks to the lab activities, the student will get specific skills in the application of the acquired knowledge of operating systems to real case-studies. **Cross-sectoral skills/soft skills** *Making judgements*: the student will acquire a critical appraisal skill in appropriately selecting technologies, algorithms and solutions according to their outcome on the performance and on the correct operation of an operating system in different application contexts. *Communication skills*: the student will learn how to describe an operating system using a technically correct and appropriate terminology. *Learning skills*: the student will learn how to be autonomous in expanding his knowledge of operating systems beyond the notions and examples learned and analyzed during the course, discovering and exploiting connections with other courses such as computer architecture, computer networks, and distributed systems.	