

<b>Principali informazioni sull'insegnamento</b>	
Titolo insegnamento	Algoritmi e Strutture Dati (corso A)
Corso di studio	Informatica L.T.
Crediti formativi	9
Denominazione inglese	Algorithms and Data Structures
Obbligo di frequenza	-
Lingua di erogazione	Italiano

<b>Docente responsabile</b>	Nome Cognome	Indirizzo Mail
	Nicola Di Mauro	nicola.dimauro@uniba.it

<b>Dettaglio crediti formativi</b>	Ambito disciplinare	SSD	Crediti
	Informatica	INF/01	9

<b>Modalità di erogazione</b>	
Periodo di erogazione	Semestre primo
Anno di corso	secondo
Modalità di erogazione	Lezioni frontali (7CFU) ed esercitazioni guidate (2CFU) 35 lezioni frontali per 86 ore (56 ore lezioni teoriche e 30 ore di esercitazioni guidate)

<b>Organizzazione e della didattica</b>	
Ore totali	139+86 = 225 corrispondenti a 9 CFU
Ore di corso	86
Ore di studio individuale	119 corrispondenti alle lezioni teoriche 20 corrispondenti alle esercitazioni guidate TOTALE 139 ore

<b>Calendario</b>	
	settembre 2020
Fine attività didattiche	gennaio 2021

<b>Syllabus</b>	
<b>Prerequisiti</b>	<ul style="list-style-type: none"> <li>- Principi della programmazione strutturata e conoscenza di almeno un linguaggio di Programmazione imperativa;</li> <li>- Principi della programmazione modulare;</li> <li>- Concetto di tipo e di dati strutturati;</li> <li>- Principi dell'astrazione funzionale (funzioni e procedure) e meccanismi di passaggio dei parametri; compilazione separata.</li> </ul>
<b>Risultati di apprendimento previsti (declinare rispetto ai Descrittori di Dublino) (si raccomanda che siano coerenti con i risultati di apprendimento del CdS, compreso i risultati di apprendimento trasversali)</b>	<ul style="list-style-type: none"> <li>• <b>Conoscenza e capacità di comprensione</b>  Il corso si propone di introdurre lo studente alle tematiche della progettazione efficiente di algoritmi mediante la metodologia dei dati astratti. Ogni struttura dati è presentata come un tipo di dato astratto al quale sono associati degli operatori di manipolazione. Questa metodologia, che prevede la specifica del tipo di dato e la rappresentazione per ogni tipo di dato astratto, sollecita lo studente a progettare proposte alternative di implementazione, dette realizzazioni, valutate ciascuna in base all'efficienza computazionale. L'analisi di algoritmi comprende la valutazione dell'impatto delle strutture dati sulla complessità dell'algoritmo e del programma. Si richiede che lo studente conosca i concetti di base della teoria della complessità e sia in grado di applicarli in semplici algoritmi di base. Accanto ai principi della astrazione, alle teorie formali del calcolo attraverso modelli algebrico-matematici, lo studente imparerà a categorizzare i problemi mediante lo strumento dello spazio di ricerca. Dovrà conoscere i paradigmi algoritmici selettivo e generativo e i problemi più noti e le tecniche risolutive che a tali paradigmi si rifanno. Lo studente avrà consapevolezza delle possibilità e dei limiti delle metodologie informatiche basate sulla programmazione orientata ad oggetti, indipendentemente dallo specifico ambiente di programmazione scelto.  Le basi devono evidenziare gli aspetti essenziali della disciplina che rimangono inalterati a fronte del cambiamento tecnologico, in modo da fornire un sistema di riferimento culturale che trascende il tempo e le circostanze.</li> <li>• <b>Conoscenza e capacità di comprensione applicate</b>  In particolare lo studente sarà in grado, da un lato, di comprendere i limiti della programmazione imperativa e, dall'altro, di cogliere le opportunità offerte dalla programmazione orientata ad oggetti e dall'uso di linguaggi appropriati. Queste competenze sono trasferite attraverso lezioni teoriche ed esercitazioni pratiche durante le quali vengono dati, per ogni tipo di dato astratto, almeno una realizzazione completa. Si richiede che lo studente, nelle ore di studio individuale, basandosi sul progetto dato a lezione, sviluppi almeno un'altra realizzazione per ogni tipo di dato. La verifica dell'acquisizione dei concetti teorici e dei metodi può essere effettuata durante l'anno accademico attraverso prove in itinere. La verifica delle abilità pratiche avviene in fase di esame con una prova di laboratorio mentre la verifica della conoscenza dei concetti di base dell'algoritmica avviene attraverso una prova scritta. Inoltre si dà l'opportunità al discente di verificare durante il corso quanto ha appreso mediante le prove in itinere</li> <li>• <b>Autonomia di giudizio</b>  Lo studente sarà in grado di valutare la qualità di una soluzione algoritmica in termini di efficienza e possibilità di riutilizzo. L'autonomia di giudizio è acquisita dai discenti attraverso i problemi posti loro a lezione e durante le esercitazioni guidate durante le quali saranno sollecitati gli studenti a lavorare in gruppi di studio e a discutere le differenti scelte realizzative. In particolare, gli studenti dovranno essere in grado di dimostrare la capacità di valutare criticamente quanto appreso, formulando un proprio punto di vista ed essendo in grado di sostenerlo nell'ambito del gruppo di lavoro, operando così in modo efficace come individuo all'interno di una squadra. Si potrà così mettere in chiaro le scelte progettuali implementate ed anche evincere i contributi personali di ogni studente partecipante al gruppo di lavoro.</li> <li>• <b>Abilità comunicative</b>  Il tipo di professionalità che implica il lavorare in gruppo e interloquire con i committenti e gli utenti finali allo scopo di comprendere le loro esigenze e rappresentare loro efficacemente i ritorni delle scelte progettuali fatte impone l'identificazione e l'acquisizione di abilità che vanno oltre le competenze tecniche. In questo corso saranno acquisite le necessarie abilità comunicative ed un'adeguata appropriatezza espressiva nella comunicazione di problematiche inerenti gli studi algoritmici, anche ad interlocutori non esperti. Queste abilità sono assicurate durante le esercitazioni pratiche in fase di progetto e realizzazione delle strutture dati, sia in autonomia che in gruppo. La</li> </ul>

Contenuti di insegnamento	<p>discussione col docente e con i colleghi del gruppo di lavoro delle possibili scelte progettuali e delle soluzioni implementative forza alla rappresentazione, alla comunicazione e alla discussione delle proprie idee.</p> <ul style="list-style-type: none"> <li>• <i>Capacità di apprendere</i></li> </ul> <p>Essendo un insegnamento del secondo anno si suppone che gli studenti abbiano già un discreto livello di autonomia nell'apprendimento e nell'approccio metodologico, capacità che consente loro di affrontare studi successivi e/o di proseguire il proprio percorso formativo con adeguata maturità, consapevoli della continua evoluzione tecnologica caratteristica della disciplina. Lo studente avrà la capacità di adattare le conoscenze acquisite anche a nuovi contesti, nonché di aggiornarsi attraverso la consultazione delle fonti specialistiche del settore algoritmico. Lo studente deve essere in grado di consultare materiale bibliografico tradizionale o reperibile via internet o attraverso piattaforme di e-learning, deve essere capace di sintetizzare quanto scritto, anche in inglese, in libri di testo e manuali e utilizzarlo in fase preparazione dell'esame. Le linee guida all'apprendimento sono date dai contenuti delle slide delle lezioni e delle esercitazioni tenute dal docente, materiale a disposizione degli studenti.</p>
	<p><b>OBIETTIVI FORMATIVI</b></p> <p>Viene presentata una metodologia di progetto formale basata sulla astrazione dei dati e sono introdotte tecniche di programmazione orientata ad oggetti. Oltre alla capacità di rappresentare in modo formale diversi tipi di problemi, vengono acquisiti i rudimenti della programmazione per classi attraverso la realizzazione di dati astratti in ambienti di programmazione orientati ad oggetti. Sono acquisiti i principi della algoritmica, presentati in funzione del modo di utilizzo dello spazio di ricerca: le caratteristiche dei paradigmi selettivo e generativo verranno evidenziate attraverso diversi algoritmi fondamentali.</p> <hr/> <p><b>OBIETTIVI PROFESSIONALIZZANTI</b></p> <p>Capacità di integrare le tecniche di astrazione funzionale con quelle di astrazione dati nella progettazione di programmi per la soluzione di un problema, individuando le strutture dati più opportune, il metodo solutivo e la tecnica algoritmica appropriata, le tecniche di realizzazione più efficienti in un linguaggio di programmazione di riferimento, valutandone i costi ed i benefici in termini di complessità di calcolo. Essere in grado di implementare diverse realizzazioni degli operatori relativi a strutture dati non primitive in un linguaggio imperativo e in uno orientato ad oggetti.</p> <p style="text-align: center;">Contenuti</p> <ol style="list-style-type: none"> <li>1. L'astrazione in programmazione</li> <li>2. Astrarre le strutture di dati: le Algebre di dati</li> <li>3. Strutture lineari di dati: liste, pile e code</li> <li>4. Cenni sulla analisi della complessità computazionale</li> <li>5. Insiemi e Dizionari</li> <li>6. Alberi binari e Alberi Binari di Ricerca</li> <li>7. Code con priorità</li> <li>8. Alberi n-ari</li> <li>9. Grafi</li> <li>10. Le tecniche algoritmiche</li> </ol> <p>11. Esercitazioni su strutture di dati e algoritmi</p>

<b>Programma</b>	
Testi di riferimento	
	<p>A. Bertossi, A. Montresor <i>Algoritmi e strutture di dati</i>. CittàStudi, 2010</p> <p>M. Cadoli, M. Lenzerini, P. Naggar, A. Schaerf . <i>Fondamenti della progettazione dei programmi</i>. Città studi Edizioni, 1997.</p> <p>C. Demetrescu, I. Finocchi, G. F. Italiano <i>Algoritmi e strutture dati 2/ed</i> Seconda edizione, McGraw-Hill, 2008.</p> <p>P. Crescenzi, G. Gambosi, R. Grossi. <i>Strutture di Dati e Algoritmi</i>, Addison-Wesley Pearson, 2/ed 2012</p>

	<p>T. Cormen, C. Leiserson, R. Rivest, C. Stein.  <i>Introduzione agli algoritmi e strutture dati</i>. Seconda edizione, McGraw-Hill, 2005.</p> <p>Clifford A. Shaffer  <i>Data Structures and Algorithm Analysis</i>  Edition 3.2 (C++Version) 2012 <a href="http://people.cs.vt.edu/~shaffer/Book">http://people.cs.vt.edu/~shaffer/Book</a></p>
Metodi didattici	<p>Lezioni frontali teoriche  Esercitazioni guidate</p>
Metodi di valutazione (indicare almeno la tipologia scritto, orale, altro)	<p>L'esame consiste in una prova di laboratorio e in una prova scritta. La prova di laboratorio è propedeutica a quella scritta.</p> <p>Le domande relative alla prova d'esame di laboratorio sono del medesimo tipo di quelle che gli studenti potranno trovare all'interno delle esercitazioni, rese loro disponibili sul sito web e sulla piattaforma ADA del dipartimento insieme alle realizzazioni in C++.</p> <p>La prova scritta finale è costituita, di norma, da domande aperte che possono riguardare sia argomenti di natura teorica che soluzioni di problemi analoghi a quelli trattati nel corso.</p>
<p>Criteria di valutazione (per ogni risultato di apprendimento atteso su indicato, descrivere cosa ci si aspetta lo studente conosca o sia in grado di fare e a quale livello al fine di dimostrare che un risultato di apprendimento è stato raggiunto e a quale livello)</p>	<p>Si richiede che lo studente sia in grado di utilizzare due formalismi: uno per la valutazione del tempo di calcolo, l'altro per la rappresentazione dei tipi di dato astratti. Lo studente deve conoscere e saper utilizzare le algebre dei dati per definire le specifiche degli operatori previsti per i diversi tipi di dato. Deve essere consapevole della indipendenza degli operatori previsti per le strutture dalle diverse realizzazioni finali che risentiranno delle specifiche scelte software e hardware effettuate. Deve essere in grado di individuare e progettare le soluzioni al problema in termini di tipo di dato astratto e di individuare la più appropriata scelta di tecnica algoritmica da perseguire anche in funzione della complessità di calcolo dell'intero programma.</p> <p>Lo studente, inoltre, dovrà dimostrare di conoscere algoritmi noti in letteratura per problemi fondamentali (ordinamento, ricerca, visite di strutture non lineari, cammini minimi su grafo, scheduling etc.) e di saperli descrivere in pseudocodice.</p> <p>Sul piano pratico, lo studente dovrà dimostrare di saper realizzare strutture dati in un ambiente di programmazione orientato ad oggetti come il C++ e di saper mettere a confronto realizzazioni differenti, valutandole anche in termini di complessità di calcolo. Deve altresì essere in grado di dimostrare come la soluzione ad un problema dato risulti indipendente dalla specifica rappresentazione adottata per il tipo di dato utilizzato e dalla tecnica realizzativa utilizzata.</p>
Altro	