Principali informazioni sull'insegnamento	**A.A. 2020-2021**
Titolo insegnamento	Algoritmi e Strutture Dati (corso B)
Corso di studio	Laurea Triennale in Informatica
Crediti formativi	9
Denominazione inglese	Algorithms and Data Structures
Obbligo di frequenza	-
Lingua di erogazione	Italiano

Docente responsabile	Nome Cognome	Indirizzo Mail
	Gianvito Pio	gianvito.pio@uniba.it
Luogo ed Orario di Ricevimento	Dip. Informatica 4° Piano – Lab KDDE	Mercoledì dalle 11:30 alle 13:30

Dettaglio credi formativi	Ambito disciplinare	SSD	Crediti
	INFORMATICO	INF-01 - Informatica	9

Modalità di erogazione	
Periodo di erogazione	Primo semestre
Anno di corso	Secondo anno
Modalità di erogazione	Lezioni frontali (7CFU) ed esercitazioni guidate (2CFU), lezioni frontali per 86 ore (56 ore lezioni teoriche e 30 ore di esercitazioni guidate)

Organizzazione della didattica	
Ore totali	225
Ore di corso	86
Ore di studio individuale	139

Calendario	
Inizio attività didattiche	5 ottobre 2020
Fine attività didattiche	13 gennaio 2021

Syllabus	
Prerequisiti	• Principi della programmazione strutturata e conoscenza di almeno un linguaggio di programmazione imperativa • Principi della programmazione modulare • Concetto di tipo e strutture dati predefinite • Principi dell'astrazione funzionale (funzioni e procedure) e meccanismi di passaggio dei parametri

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)	• *Conoscenza e capacità di comprensione* 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 dell'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. • *Conoscenza e capacità di comprensione applicate* 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 forniti, per ogni tipo di dato astratto, almeno una realizzazione completa e problemi con relativa soluzione. Si richiede che lo studente, nelle ore di studio individuale, basandosi sugli esercizi dati a lezione, sviluppi almeno un'altra realizzazione per ogni tipo di dato. 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. • *Autonomia di giudizio* 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 alla soluzione proposta. • *Abilità comunicative* 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 capacità 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 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. • *Capacità di apprendere* 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 durante la 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.
Contenuti di insegnamento	OBIETTIVI FORMATIVI Viene presentata una metodologia di progetto formale basata sull'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. OBIETTIVI PROFESSIONALIZZANTI 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 risolutivo 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. Abilità nell'implementare diverse realizzazioni degli operatori relativi a strutture dati non primitive in un linguaggio imperativo e in uno orientato ad oggetti. CONTENUTI 1. Algoritmi e programmi Il ruolo delle tecniche di astrazione nel progetto di programmi. Dall'algoritmo al programma; la valutazione dell'algoritmo. 2. Algebre di dati Dati e rappresentazioni, requisiti delle astrazioni di dati, costrutti. Astrazioni di dati e dati primitivi. Algebre di dati: specifica sintattica e semantica. La realizzazione. 3. Strutture lineari di dati Liste: specifiche, realizzazioni attraverso rappresentazioni sequenziali e collegate. Pile: specifiche e realizzazioni alternative, pile e procedure ricorsive. Code: specifiche e realizzazioni alternative. Scelta, implementazione e verifica di algoritmi per la ricerca, l'ordinamento e la fusione delle strutture dati proposte. 4. Insiemi e Dizionari Insiemi: specifiche e confronto tra realizzazioni alternative. Dizionari: specifiche e confronto di realizzazioni alternative (Hash chiuso e aperto, vettori e liste ordinate). 5. Strutture non lineari di dati: Alberi binari ed n-ari. Generalità. Gli alberi radicati e alberi ordinati, proprietà. Alberi binari: specifiche, definizione ricorsiva, la corrispondenza con le liste, rappresentazioni e realizzazioni. Alberi binari di ricerca, alberi bilanciati. Alberi n-ari: specifiche, definizione ricorsiva, rappresentazioni e realizzazioni alternative. Algoritmi su alberi binari ed n-ari: visita, inserimento di un valore e ricerca di un valore. 6. Code con priorità Generalità. Specifiche, rappresentazioni e realizzazioni alternative.

	7. Strutture non lineari di dati: Grafi Il tipo astratto GRAFO: specifiche sintattiche e semantiche. Rappresentazioni mediante matrici di adiacenza e di incidenza, vettori di adiacenza, liste di adiacenza, e rappresentazioni collegate: realizzazioni alternative. Algoritmi su grafi: visita di un grafo, cammini minimi e generazione del minimo albero di copertura. 8. Analisi della Complessità di calcolo Generalità. Tempo di calcolo. Ordini di grandezza e complessità. Modelli di costo per la complessità in tempo e in spazio. Definizione delle funzioni di costo. 9. Le tecniche algoritmiche Classificazione dei problemi: problemi di ricerca, di decisione, di ottimizzazione. Lo spazio di ricerca: definizione e proprietà. Le tecniche algoritmiche: il paradigma selettivo e il paradigma generativo. Tecnica dell'enumerazione, del backtracking, la tecnica greedy, la tecnica divide-et-impera. Problemi e metodi solutivi: string matching (Knuth-Morris-Pratt), partizionamento di insiemi, problema delle N-Regine, problema dello zaino, problema del commesso viaggiatore, problema della colorazione, ricerca del percorso più breve in un grafo (Dijkstra), minimo albero di copertura (Kruskal), selezione di attività.

Programma	
Testi di riferimento	A. Bertossi, A. Montresor. *Algoritmi e strutture di dati*. Città Studi, 2010 M. Cadoli, M. Lenzerini, P. Naggar, A. Schaerf . *Fondamenti della progettazione dei programmi*. Città studi Edizioni, 1997. C. Demetrescu, I. Finocchi, G. F. Italiano. *Algoritmi e strutture dati 2/ed Seconda edizione*, McGraw-Hill, 2008. P. Crescenzi, G. Gambosi, R. Grossi. *Strutture di Dati e Algoritmi*, Addison-Wesley Pearson, 2/ed 2012 T. Cormen, C. Leiserson, R. Rivest, C. Stein. *Introduzione agli algoritmi e strutture dati*. Seconda edizione, McGraw-Hill, 2005. Clifford A. Shaffer. *Data Structures and Algorithm Analysis*. Edition 3.2 (C++Version) 2012 http://people.cs.vt.edu/~shaffer/Book
Note ai testi di riferimento	I libri di testo sono integrati con le slide e le dispense del docente.
Metodi didattici	Le lezioni frontali saranno dedicate all'apprendimento dei modelli teorici e dei concetti di base coadiuvati da alcuni esempi. Le ore di esercitazione saranno dedicate sia all'esecuzione di esercizi in classe, anche coinvolgendo direttamente gli studenti nella risoluzione degli stessi, sia alla realizzazione delle strutture dati e delle tecniche di programmazione in un linguaggio orientato agli oggetti. Si prevede l'utilizzo della piattaforma di e-learning del dipartimento per la pubblicazione del materiale didattico, la

	discussione degli argomenti delle lezioni tra docente/studente e studenti/studenti, la condivisione dei risultati di laboratorio, la condivisione degli esercizi e la pubblicazione di materiale integrativo e di approfondimento.
Metodi di valutazione (indicare almeno la tipologia scritto, orale, altro)	L'esame consiste in una prova di laboratorio e in una prova scritta. La prova scritta è propedeutica a quella di laboratorio. 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 di e-learning del dipartimento insieme alle realizzazioni in C++. La prova scritta finale è costituita da domande aperte che possono riguardare sia argomenti di natura teorica che lo sviluppo di una soluzione a problemi analoghi a quelli trattati durante il corso.
Criteri 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)	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. 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 realizzare o, quanto meno, descriverli in pseudocodice. Sul piano pratico, lo studente dovrà dimostrare di saper realizzare strutture dati in un ambiente di programmazione orientato agli 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.
Altro	