| **Principali informazioni sull'insegnamento** | | |
|---|---|---|
| Denominazione dell'insegnamento | ***Algoritmi e Strutture Dati (corso A)*** | |
| Corso di studio | *Informatica L.T.* | |
| Anno di corso | *II* | |
| Crediti formativi universitari (CFU) / European Credit Transfer and Accumulation System (ECTS): | | 9 |
| SSD | *INF-01* | |
| Lingua di erogazione | *Italiano* | |
| Periodo di erogazione | *Settembre 2021 – Gennaio 2022* | |
| Obbligo di frequenza | *Fortemente consigliato* | |

| **Docente** | |
|---|---|
| Nome e cognome | Nicola Di Mauro |
| Indirizzo mail | nicola.dimauro@uniba.it |
| Telefono | 080 5442297 |
| Sede | *Dipartimento di Informatica, Università degli Studi di Bari* |
| Sede virtuale | - |
| Ricevimento (giorni, orari e modalità) | Lunedì 10-12 o altro giorno previa prenotazione per email |

| ***Syllabus*** | |
|---|---|
| **Obiettivi formativi** | Viene presentata una metodologia di progetto formale basata sulla astrazione dei dati e sono introdotte tecniche di programmazione orientata ad oggetti. Oltre alla capacita' 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.<br><br>**Obiettivi Professionalizzanti**<br><br>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. |
| **Prerequisiti** | - Principi della programmazione strutturata e conoscenza di almeno un linguaggio di programmazione imperativa<br>- Principi della programmazione modulare<br>- Concetto di tipo e strutture dati predefinite<br>- Principi dell'astrazione funzionale (funzioni e procedure) e meccanismi di passaggio dei parametri |
| **Contenuti di insegnamento (Programma)** | 1. Algoritmi e programmi<br>Il ruolo delle tecniche di astrazione nel progetto di programmi.<br>Dall'algoritmo al programma; la valutazione dell'algoritmo. |

| | |
|---|---|
| | 2. Algebre di dati<br>Dati e rappresentazioni, requisiti delle astrazioni di dati, costrutti. Astrazioni di dati e dati primitivi. Algebre di dati: specifica sintattica e semantica. La realizzazione.<br><br>3. Strutture lineari di dati<br>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.<br><br>4. Insiemi e Dizionari<br>Insiemi: specifiche e confronto tra realizzazioni alternative. Dizionari: specifiche e confronto di realizzazioni alternative (Hash chiuso e aperto, vettori e liste ordinate).<br><br>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.<br><br>6. Code con priorità<br>Generalità. Specifiche, rappresentazioni e realizzazioni alternative.<br><br>7. Strutture non lineari di dati: Grafi<br>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.<br><br>8. Analisi della Complessità di calcolo<br>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.<br><br>9. Le tecniche algoritmiche<br>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. |
| **Testi di riferimento** | *A. Bertossi, A. Montresor. Algoritmi e strutture di dati. Città Studi, 2010*<br><br>*M. Cadoli, M. Lenzerini, P. Naggar, A. Schaerf . Fondamenti della progettazione* |

| | |
|---|---|
| | *dei programmi. Città studi Edizioni, 1997.*<br><br>*C. Demetrescu, I. Finocchi, G. F. Italiano. Algoritmi e strutture dati 2/ed Seconda edizione, McGraw-Hill, 2008.*<br><br>*P. Crescenzi, G. Gambosi, R. Grossi. Strutture di Dati e Algoritmi, Addison-Wesley Pearson, 2/ed 2012*<br><br>*T. Cormen, C. Leiserson, R. Rivest, C. Stein. Introduzione agli algoritmi e strutture dati. Seconda edizione, McGraw-Hill, 2005.*<br><br>*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. |

| **Organizzazione della didattica** | | | |
|---|---|---|---|
| **Ore** | | | |
| Totali | Didattica frontale | Pratica (laboratorio, campo, esercitazione, altro) | Studio individuale |
| *225* | *56* | *30* | *139* |
| **CFU/ETCS** | | | |
| *9* | *7* | *2* | |

| **Metodi didattici** | |
|---|---|
| | Le lezioni frontali saranno dedicate all'apprendimento dei modelli teorici e dei concetti di base coadiuvati da alcuni esempi.<br>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.<br>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. |

| **Risultati di apprendimento previsti** | |
|---|---|
| **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.<br><br>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** | 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. |
| **Competenze trasversali** | Autonomia di giudizio<br><br>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.<br><br>Abilità comunicative<br><br>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.<br><br>Capacità di apprendere<br><br>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. |

| **Valutazione** | |
|---|---|
| Modalità di verifica dell'apprendimento | L'esame consiste in una prova di laboratorio e in una prova scritta. La prova scritta è propedeutica a quella di laboratorio.<br>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++.<br>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 | 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. |
| Criteri di misurazione dell'apprendimento e di attribuzione del voto finale | Il voto finale è attribuito in trentesimi. L'esame si intende superato quando il voto è maggiore o uguale a 18. L'accesso alla prova scritta è consentito se si è conseguito un voto maggiore o uguale a 18 alla prova di laboratorio. |
| **Altro** | |
| | |

| **General information** | | |
|---|---|---|
| Academic subject | Algorithms and Data Structures | |
| Degree course | Laurea Triennale in Informatica | |
| Academic Year | *II* | |
| European Credit Transfer and Accumulation System (ECTS) | | 9 |
| Language | *Italian* | |
| Academic calendar (starting and ending date) | *September 2021 – January 2022* | |
| Attendance | | |

| **Professor/ Lecturer** | |
|---|---|
| Name and Surname | Nicola Di Mauro |
| E-mail | nicola.dimauro@uniba.it |
| Telephone | 080 5442297 |
| Department and address | *Conmputer Science, University of Bari* |
| Virtual headquarters | |
| Tutoring (time and day) | Monday 10-12 by email appointment |

| ***Syllabus*** | |
|---|---|
| **Learning Objectives** | A formal design methodology, based on data abstraction, is presented and object-oriented programming techniques are introduced. In addition to the ability to formally represent different types of problems, the basics of object-oriented programming are acquired through the realization of abstract data structures. The principles of algorithmics are acquired, presented according to the way the search space is explored: the characteristics of the selective and generative paradigms will be highlighted through several fundamental algorithms. The students will gain the ability to integrate functional abstraction techniques with data abstraction techniques in the design of programs for the solution of a problem, identifying the most appropriate data structures, the method and the appropriate algorithmic technique, the most efficient realization techniques in a reference language of programming, evaluating the costs and benefits in terms of computational complexity. Ability to implement different realizations of operators related to non-primitive data structures in an imperative and in an object-oriented programming language. |
| **Course prerequisites** | - Principles of structured programming and knowledge of at least one imperative programming language<br>- Principles of modular programming<br>- Concept of "type" and predefined data structures<br>- Principles of functional abstraction (functions and procedures) and |

| | |
|---|---|
| | mechanisms for parameter passing |
| **Contents** | 1. Algorithms and programs<br>The role of abstraction techniques in the design of programs. From the algorithm to the program; the evaluation of the algorithm.<br><br>2. Data algebras<br>Data and representations, requirements of data abstractions, constructs. Data abstractions and primitive data. Data algebras: syntactic and semantic specification. The realization.<br><br>3. Linear data structures<br>Lists: specifications, realizations through sequential and linked representations. Stacks: specifications and alternative realizations, stacks and recursive procedures. Queue: specifications and alternative realizations. Selection, implementation and verification of algorithms for searching, sorting and merging the proposed data structures.<br><br>4. Sets and Dictionaries<br>Sets: specifications and comparison between alternative realizations. Dictionaries: specifications and comparison of alternative realizations (closed and open hash, vectors and ordered lists).<br><br>5. Nonlinear data structures<br>Binary and n-ary trees. Principles. Rooted trees and sorted trees, properties. Binary trees: specifications, recursive definition, correspondence with lists, representations and realizations. Binary search trees,<br>balanced trees. N-ary trees: specifications, recursive definition, alternative representations and realizations. Algorithms on binary and n-ary trees: visit, insertion of a value and search for a value.<br><br>6. Priority queues<br>Basics. Specifications, representations and alternative implementations.<br><br>7. Non-linear data structures: Graphs<br>The abstract type GRAPH: syntactic and semantic specifications. Representations using adjacency and incidence matrices, adjacency vectors, adjacency lists, and related representations: alternative realizations. Algorithms on graphs: visit of a graph, shortest paths and generation of the minimum spanning tree.<br><br>8. Analysis of the Computational Complexity<br>Basics. Time of computation. Orders of magnitude and complexity. Cost models for complexity in time and space. Definition of cost functions.<br><br>9. Algorithmic techniques<br>Problem classification: research, decision, optimization problems. The research space: definition and properties. Algorithmic techniques: the selective paradigm and the generative paradigm. Technique of enumeration, backtracking, the greedy technique, and the divide-et-impera technique. Problems and solutions: string matching (Knuth-Morris-Pratt), partitioning of sets, N-Queens problem, knapsack problem, traveling salesman problem, |

| | |
|---|---|
| | coloring problem, search for the shortest path in a graph (Dijkstra), minimum spanning tree (Kruskal), selection of activities. |
| **Books and bibliography** | *A. Bertossi, A. Montresor. Algoritmi e strutture di dati. Città Studi, 2010*<br><br>*M. Cadoli, M. Lenzerini, P. Naggar, A. Schaerf . Fondamenti della progettazione dei programmi. Città studi Edizioni, 1997.*<br><br>*C. Demetrescu, I. Finocchi, G. F. Italiano. Algoritmi e strutture dati 2/ed Seconda edizione, McGraw-Hill, 2008.*<br><br>*P. Crescenzi, G. Gambosi, R. Grossi. Strutture di Dati e Algoritmi, Addison-Wesley Pearson, 2/ed 2012*<br><br>*T. Cormen, C. Leiserson, R. Rivest, C. Stein. Introduzione agli algoritmi e strutture dati. Seconda edizione, McGraw-Hill, 2005.*<br><br>*Clifford A. Shaffer. Data Structures and Algorithm Analysis. Edition 3.2 (C++Version) 2012 http://people.cs.vt.edu/~shaffer/Book* |
| **Additional materials** | Books will be integrated by the slides provided by the teacher. |

| **Work schedule** | | | |
|---|---|---|---|
| Total | Lectures | Hands on (Laboratory, working groups, seminars, field trips) | Out-of-class study hours/ Self-study hours |
| **Hours** | | | |
| *225* | *56* | *30* | *139* |
| **ECTS** | | | |
| *9* | *7* | *2* | |

| | |
|---|---|
| **Teaching strategy** | |
| | The lectures will be dedicated to learning theoretical models and basic concepts supported by some examples. The hours of practice will be dedicated to running exercises in the classroom, also involving the students directly in solving them, to the realization of data structures and to programming techniques in an object-oriented language.<br>The use of the e-learning platform of the department is foreseen for the publication of the teaching material, the discussion of the topics of the lessons between the teacher / student and students / students, the sharing of laboratory results, the sharing of exercises and the publication of supplementary and in-depth material. |
| **Expected learning outcomes** | |
| **Knowledge and understanding on:** | The course aims to introduce the student to the topic of efficient algorithm design using the abstract data methodology. Each data structure is presented as an abstract data type to which manipulation operators are associated. This methodology, which provides for the specification of the data type and the representation for each type of abstract data, pushes the student to design alternative implementation proposals, called realizations, each evaluated on the basis of computational efficiency. The analysis of algorithms includes the evaluation of the impact of data structures on the complexity of the algorithm |

| | |
|---|---|
| | and the program. The student is required to know the basic concepts of complexity theory and be able to apply them in simple basic algorithms.<br>Alongside the principles of abstraction, the formal theory of calculus through algebraic-mathematical models, the student will learn to categorize problems using the research space. The student will have to know the selective and generative algorithmic paradigms and the best-known problems and the solution techniques that refer to these paradigms. The student will be aware of the possibilities and limitations of computer methodologies based on object-oriented programming, regardless of the specific programming environment chosen. The foundations must highlight the essential aspects of the discipline that remain unchanged despite the technological evolution, in order to provide a cultural reference system independently on time and circumstances. |
| **Applying knowledge and understanding on:** | The student will be able, on the one hand, to understand the limits of imperative programming and, on the other, to seize the opportunities offered by object-oriented programming and the use of appropriate languages.<br>These skills are transferred through theoretical lessons and practical exercises during which, for each type of abstract data, at least one complete realization and problems with relative solution are provided. It is required that the student, in the hours of individual study, based on the exercises given in class, develops at least one other realization for each type of data.<br>The verification of practical skills takes place during the examination phase with a laboratory test while the verification of knowledge of the basic concepts of the algorithm takes place through a written test. |
| **Soft skills** | Making informed judgments and choices<br>The student will be able to evaluate the quality of an algorithmic solution in terms<br>of efficiency and possibility of reuse. Autonomy of judgment is acquired by the students through the problems posed to them in the lectures and during the guided exercises, during which the students will be encouraged to work in study<br><br>groups and to discuss the different implementation choices. In particular, students<br>must be able to demonstrate the ability to critically evaluate what they have learned, formulating their own point of view and being able to support it within the working group, thus operating effectively as an individual within a team. It will<br>thus be possible to clarify the design choices implemented and also to deduce the<br>personal contributions of each student to the proposed solution.<br><br>Communication skills<br>The type of professionalism, which implies working in groups and interacting with<br>clients and end users in order to understand their needs and effectively represent |

| | |
|---|---|
| | to them the returns of the design choices made, requires the identification and acquisition of skills that go beyond technical skills. In this course the necessary communication skills and an adequate expressive ability in communicating problems related to algorithmic studies will be acquired, even to non-expert interlocutors. These skills are ensured during practical exercises in the design and implementation phase of the data structures, both independently and in groups.<br>The discussion with the teacher and with the colleagues of the working group of the possible design choices and implementation solutions forces the representation, communication and discussion of own ideas.<br><br>Learning skills<br>Being a second-year course, it is assumed that students already have a fair level of autonomy in learning and in the methodological approach, an ability that allows them to face subsequent studies and / or to continue their training with adequate maturity, aware of the continuous technological evolution characteristic of the discipline. The student will have the ability to adapt the acquired knowledge also to new contexts, as well as to keep themselves updated by consulting the specialized sources of the algorithmic sector. The student must be able to consult traditional bibliographic material or available via the internet or through e-learning platforms, must be able to summarize what is written, also in English, in textbooks and manuals and use it during exam preparation. The learning guidelines are given by the contents of the slides of the lessons and exercises held by the teacher, that is made available to students. |
| **Assessment and feedback** | |
| Methods of assessment | The exam consists of a laboratory test and a written test. Passing the written test is mandatory before the laboratory test.<br>The questions relating to the laboratory exam are of the same type as those that students will be able to find in the exercises, made available to them on the website and on the e-learning platform of the department together with the realizations in C ++.<br>The final written test consists of open questions that can cover both theoretical topics and the development of a solution to problems similar to those dealt with during the course. |
| Evaluation criteria | The student is required to be able to use two formalisms: one for the evaluation |

| | |
|---|---|
| | of<br>the computational time, the other for the representation of abstract data types.<br><br>The student must know and be able to use data algebras to define the specifications of the operators provided for the different types of data. The student must be aware of the independence of the operators foreseen for the<br>structures from the various final realizations that will be affected by the specific software and hardware choices made. The student must be able to identify and design solutions to the problem in terms of abstract data type and to identify the<br>most appropriate choice of algorithmic technique to be pursued also according to<br>the computational complexity of the entire program. Furthermore, the student will have to demonstrate knowledge of algorithms known in the literature for fundamental problems (sorting, research, visit of non-<br>linear structures, shortest paths on graphs, scheduling, etc.) and to be able to implement them or, at least, to describe them in pseudocode. On a practical level, the student will have to demonstrate that is able to create data structures in an object-oriented programming environment such as C++ and<br>to be able to compare different realizations, also evaluating them in terms of computational complexity. The student must also be able to demonstrate how the<br>solution to a given problem is independent of the specific representation adopted<br>for the type of data used and the used realization technique. |
| Criteria for assessment and attribution of the final mark | The final grade range is between 1 and 30. The exam is passed when the grade is<br>greater than or equal to 18. The access to the written test is allowed if a grade greater than or equal to 18 is achieved in the laboratory test. |
| **Additional information** | |
| | |
| | |
| | |
| | |