Principali informazioni sull'insegnamento

Denominazione dell'insegnamento	**Programmazione (track E-M)**
Corso di studio	Informatica
Anno Accademico	2025/26
Crediti formativi universitari (CFU) / European Credit Transfer and Accumulation System (ECTS)	12 CFU
Settore Scientifico Disciplinare	INF/01
Lingua di erogazione	Italiano
Anno di corso	Primo
Periodo di erogazione	1^ semestre, le date esatte sono riportate nel manifesto/regolamento
Obbligo di frequenza	No, ma la frequenza è fortemente raccomandata
Sito web del corso di studio	https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea

Docente/i

Nome e cognome	VERONICA ROSSANO
Indirizzo mail	veronica.rossano@uniba.it
Telefono	080 544 2477
Sede	Dipartimento di Informatica, Via Orabona 4, 70125, Bari. Stanza n.772, 7^ piano.
Sede virtuale	Piattaforma e-learning UNIBA - https://elearning.uniba.it/course/view.php?id=2830
Sito web del docente	https://www.uniba.it/it/docenti/rossano-veronica
Ricevimento (giorni, orari e modalità, es. su appuntamento)	Tutti i giorni previo appuntamento via mail o su MsTeams.

Syllabus

<table>
<tr><td>Obiettivi formativi</td><td>Il corso si propone di introdurre la metodologia del problem solving e gli elementi base della programmazione imperativa strutturata per formulare soluzioni algoritmiche a problemi di varia complessità. In particolare, lo studente acquisirà la capacità di applicare le tecniche del problem solving, di individuare strategie di soluzione di natura algoritmica e di usare il linguaggio di programmazione C per implementarle.</td></tr>
<tr><td>Prerequisiti</td><td>Per frequentare il Corso di Laurea in Informatica non si richiedono competenze informatiche di alcun tipo, ma è indispensabile avere una buona preparazione nelle materie di base della scuola media secondaria, in particolare si richiedono abilità matematiche, logiche e di ragionamento.</td></tr>
<tr><td>Contenuti di insegnamento (Programma)</td><td>
<table>
<tr><th>Mod</th><th>Argomenti</th><th>Impegno orario stimato</th></tr>
<tr><td>1</td><td>Presentazione del corso, contenuti, modalità d'esame, esoneri, frequenza alle lezioni, orari, modalità di esercitazione in aula, ecc.
Introduzione alla Programmazione: dal problema al programma</td><td>3</td></tr>
<tr><td>2</td><td>L’algoritmo come astrazione.
Le proprietà dell’algoritmo</td><td>3</td></tr>
<tr><td>3</td><td>Il Problem Solving e la fase di analisi
L’analisi del problema. La formulazione del problema e le sue specifiche. I dati del problema. I risultati attesi. Gli errori e i casi
limite. Ambiguità, limiti dei valori e controlli dei dati.
Esempi ed esercizi.</td><td>6</td></tr>
<tr><td>4</td><td>Il Problem Solving e la fase di progettazione
La scomposizione del problema in sottoproblemi. Le tecniche: top-down, bottom-up, ibrida.
La costruzione del metodo solutivo. La rappresentazione dell’algoritmo mediante linguaggi di rappresentazione: i diagrammi di flusso strutturati, il linguaggio lineare, l’albero di decomposizione, i diagrammi di Nassi-Schneidermann. Il teorema di Bohm-Jacopini: enunciato.
Esempi ed esercizi.</td><td>15</td></tr>
<tr><td>5</td><td>Problemi semplici e problemi complessi. Scomposizione sequenziale, selettiva, iterativa e ricorsiva.
Esempi ed esercizi.</td><td>3</td></tr>
<tr><td>6</td><td>Il programma. Le variabili. Tipi di istruzione. Istruzioni dichiarative.
Istruzioni di ingresso-uscita. Istruzione di assegnazione. Espressioni aritmetiche. Espressioni logiche. Costanti. Strutture di controllo.
Esempi ed esercizi.</td><td>3</td></tr>
<tr><td>7</td><td>Algoritmi elementari: conteggio, sommatoria e media di un insieme di
numeri, fattoriale, conversione da caratteri a numeri in base 10, numero primo, massimo comun divisore, serie di Fibonacci, scambio.</td><td>3</td></tr>
<tr><td>8</td><td>Tipi di dati. La dichiarazione di tipo. Tassonomia dei tipi di dato. Tipi
standard. Tipi semplici definiti dall’utente: enumerativo, subrange.</td><td>3</td></tr>
<tr><td>9</td><td>Tipi di dati. Tipi strutturati. Dato strutturato e tipo Array. Struttura di dati e tipo Record. Rappresentazione interna degli array e dei record. Gli indici degli array.
Esempi ed esercizi.</td><td>7</td></tr>
<tr><td>10</td><td>Esercitazione e preparazione alla prova di autovalutazione: analisi di problemi, scomposizione in sottoproblemi, progettazione algoritmica della soluzione. Attività partecipativa cooperativa svolta con l’assistenza del docente</td><td>3</td></tr>
</table>
</td></tr>
</table>

11	Esercitazione per autovalutazione: risoluzione di un problema secondo le fasi del problem solving escluso la codifica.	*4*
12	Auto correzione prova di autovalutazione assistita dal docente	*2*
13	La programmazione modulare. I sottoprogrammi. Utilità dei sottoprogrammi. Sottoprogramma come astrazione funzionale. Struttura risultante di un programma. La chiamata dei sottoprogrammi. Nidificazione. Esempi.	*8*
14	La comunicazione dei sottoprogrammi con l'ambiente esterno e l'ambiente chiamante. La vista di un sottoprogramma. L'effetto shadowing. Variabili globali, locali e non locali. Le regole di visibilità. L'ambito e la durata delle variabili. Esempi ed esercizi.	*3*
15	I sottoprogrammi. Allocazione statica e allocazione dinamica. Il side effect. Il contour model. Esempi.	*3*
16	Sottoprogrammi. I parametri effettivi e i parametri formali. Modalità di passaggio dei parametri: per valore, per referenza e per nome. Valutazione e confronto tra le modalità di passaggio. Esempi ed esercizi.	*4*
17	Sottoprogrammi. Le procedure. Intestazione e chiamata delle procedure. Esempi ed esercizi.	*3*
18	Sottoprogrammi. Le funzioni. Intestazione e chiamata delle funzioni. Procedure vs funzioni. Esempi ed esercizi.	*3*
19	Sottoprogrammi. Sottoprogrammi come parametri. Attivazione dei sottoprogrammi. I record di attivazione e lo stack. La concatenazione statica e la concatenazione dinamica. La gestione dell'esecuzione. Esempi ed esercizi.	*4*
20	La ricorsione. Funzioni ricorsive. Gestione dell'esecuzione con le funzioni ricorsive. Esempi ed esercizi.	*3*
21	Algoritmi fondamentali: Algoritmi su array: stampa istogrammi mediante array, elimina duplicati su array ordinati. Algoritmi su matrici. Ricerca lineare. Ricerca binaria. Fusione di array	*3*
22	Algoritmi di ordinamento: sort per selezione, per scambio, per inserzione. Algoritmi ricorsivi: Fattoriale.	*3*
23	Struttura dei programmi C. Tipi di dati: semplici predefiniti – int, float, double e char-, enumerazione esplicita dei valori. Definizione di tipo.	*3*
24	Tipi strutturati in C: il costruttore array, il costruttore struct ed il costruttore puntatore. Compatibilità dei tipi.	*4*
25	Strutture di controllo in C. Istruzioni di selezione: If, Switch; Istruzioni iterative: while, do-while, for.	*3*
26	Funzioni e procedure in C: definizione, chiamata, prototipo, passaggio dei parametri.	*3*
27	Ambito di visibilità delle variabili in C. Array come parametri. Strutture come parametri.	*4*
28	Effetti collaterali. Procedure e funzioni predefinite. Standard library. Puntatori. File.	*5*

Testi di riferimento	**Testo da cui studiare:** P. Deitel e H. Deitel Il linguaggio C – Fondamenti e tecniche di programmazione 8^edizione - Pearson 2016 - ISBN: 9788891901651 (vanno bene anche le edizioni successive e precedenti dalla 4^ in poi) **Testo integrativo, facoltativo:** W. B. Kernighan, D.M. Ritchie. Il linguaggio C. Principi di programmazione e manuale di riferimento. Gli studenti che lo desiderano possono ottenere i testi in prestito dalla Biblioteca. Può convenire verificarne la disponibilità mediante il Sistema Bibliotecario di Ateneo https://opac.uniba.it/easyweb/w8018/index.php? e contattare la biblioteca per concordare il prestito.
Note ai testi di riferimento	Nel corso delle lezioni il docente illustrerà i concetti con l'ausilio di slide che sintetizzano i contenuti del corso. Le slide saranno rese disponibili al termine di ogni lezione sulla piattaforma ADA del dipartimento (v. sopra 'sede virtuale'). Sulla piattaforma ADA sono disponibili, inoltre, tutte alcune simulazioni di esame già svolte.

Organizzazione della didattica

Ore

Totali	Didattica frontale	Laboratorio/esercitazione	Progetto	Studio individuale
300 ore	72 ore	45 ore		183 ore

CFU/ETCS				
12 CFU	9 CFU	3 CFU		

Metodi didattici

Il corso è organizzato in lezioni frontali svolte con l'ausilio di slide, in esercitazioni guidate svolte in aula e esercitazioni svolte autonomamente a casa.

Le esercitazioni guidate saranno svolte in aula con l'approccio del Bring Your Own Device (BYOD).

Le esercitazioni svolte in aula e a casa dovranno essere svolte singolarmente da ciascuno studente. Sarà richiesta la consegna delle stesse sulla piattaforma ADA seguendo le indicazioni e le scadenze comunicate durante le lezioni frontali. Le esercitazioni consegnate saranno utili al docente per verificare la partecipazione alle esercitazioni e la comprensione degli argomenti svolti a lezione.

Agli studenti non frequentanti non è richiesta la sottomissione delle esercitazioni.

Risultati di apprendimento previsti	*Nella sezione **Risultati di apprendimento previsti** è necessario riportare **sinteticamente** enunciazioni generali dei tipici risultati conseguiti dagli studenti in ogni insegnamento; definiscono pertanto quali sono i risultati dell'apprendimento. Non vanno intesi come prescrizioni; non rappresentano soglie o requisiti minimi e non sono esaustivi; i **descrittori** mirano a identificare la natura del dell'insegnamento nel suo complesso; non hanno carattere disciplinare e non sono circoscritti in determinate aree accademiche o professionali.* *Attenzione sono DIVERSI per ogni ciclo di studi, dalla triennale al dottorato. Qui ne trovate una descrizione esaustiva: https://www.unipi.it/index.php/qualita-didattica/item/21031-descrittori-di-dublino*
Conoscenza e capacità di comprensione	• Acquisire la conoscenza degli aspetti teorici e pratici relativi alla progettazione delle soluzioni di problemi (problem solving) mediante l'uso del computer e la conoscenza degli strumenti che si utilizzano per la programmazione. • Acquisire conoscenze che consentano allo studente di comprendere come si può indicare ad un elaboratore elettronico (macchina automatica di impiego universale, hardware) la soluzione di un problema o di una classe di problemi, che l'elaboratore può risolvere, con un metodo ed un linguaggio appropriato, creando un apposito programma (software) eseguibile dall'elaboratore. • Acquisire la capacità di ragionare ed individuare una soluzione ad un problema (algoritmo) secondo il paradigma della programmazione imperativa strutturata. • Acquisire autonomia di studio nella disciplina e vocabolario tecnico.
Conoscenza e capacità di comprensione applicate	• Comprendere l'uso di un linguaggio di progettazione non convenzionale (es. pseudocodice) e l'uso di una rappresentazione grafica (es. flow chart) per descrivere con un formalismo semplice un algoritmo; • Comprendere il lessico, la sintassi e la semantica del linguaggio di programmazione C; • Acquisire la capacità di scrivere un programma strutturato in linguaggio C; • Acquisire la capacità di individuare casi di test per il dominio cui fa riferimento il programma creato; • Acquisire la capacità di utilizzare un ambiente di sviluppo per trasformare il programma sorgente (in C) in programma eseguibile ed eseguirlo.
Competenze trasversali	**Autonomia di giudizio** • *Acquisire la capacità di verificare che l'algoritmo individuato risponda alle specifiche di un problema;* • *Saper valutare e interpretare in maniera autonoma diverse strategie risolutive analizzando gli algoritmi proposti e fornendo soluzioni alternative.* • *Acquisire la capacità di verificare che i risultati ottenuti dopo l'esecuzione del programma siano quelli attesi.* **Abilità comunicative** • *Imparare a commentare il codice prodotto al fine di renderlo comprensibile e agevolmente modificabile da altri professionisti, con l'obiettivo di sviluppare in team.*

	Capacità di apprendere in modo autonomo • *Capacità di approfondire concetti attraverso lo studio autonomo di materiale* • *didattico bibliografico anche attraverso piattaforme di e-learning.* • *Capacità di completare autonomamente il percorso formativo previsto dal testo di riferimento, oltre i contenuti previsti dal programma dell'insegnamento.* • *Capacità di riutilizzare le conoscenze acquisite sia in situazioni problematiche nuove, sia in contesti nuovi di programmazione imperativa.*

Valutazione	
Modalità di verifica dell'apprendimento	**Valutazione Formativa (utile per monitorare il raggiungimento degli obiettivi)** • Esercitazioni da svolgere in itinere (**non obbligatorie**). • Test a risposta multipla proposti durante le lezioni. Le esercitazioni e i test non saranno soggetti a valutazione ma saranno utilizzati per verificare il raggiungimento degli obiettivi. **Valutazione Sommativa (Esame):** • *Prova scritta/di laboratorio*: la prova si svolge nei laboratori di informatica e consiste nella soluzione di un problema complesso. La prova ha durata di circa 3 ore ed è richiesto di applicare tutte le fasi del problem solving, dall'analisi passando per la progettazione e terminando con la codifica in linguaggio C. La prova scritta è propedeutica alla prova orale e si considera superata con un punteggio minimo di 16/30. • *Prova orale*: consiste nella visione della prova scritta e discussione degli eventuali errori commessi. Durante la prova orale saranno poste due riguardanti la parte teorica trattata durante il corso. La *prova esonerante*, in forma scritta e articolata in domande a risposta aperta ed esercizi di programmazione in C, si terrà al termine di un numero congruo di argomenti teorici. Sarà valutata in trentesimi, si ritiene superata se il suo voto è maggiore di 18 e il suo superamento esonera lo studente dallo svolgimento **della prova orale**. I voti di questa prova esonerante possono essere utilizzati esclusivamente fino a **Settembre**. Lo studente può, se preferisce, rinunciare al voto della prova esonerante e partecipare alla prova orale. Il voto finale conseguito è calcolato come media dei voti ottenuti alle due prove (prova di laboratorio e prova orale/prova esonerante) e viene comunicato tramite piattaforma Esse3.
Criteri di valutazione	**Conoscenza e capacità di comprensione:** • Lo studente dovrà essere in grado di analizzare problemi formulando anche ipotesi aggiuntive, individuando i dati necessari e sufficienti per la soluzione e fornendone la descrizione. • Dovrà essere in grado di individuare una strategia di soluzione che prevede la scomposizione del problema in sottoproblemi e di saper rappresentare sia la scomposizione sia gli algoritmi con adeguati linguaggi di descrizione.

<table>
<tr><td></td><td>

- Dovrà dimostrare di saper implementare la soluzione proposta utilizzando il linguaggio imperativo di riferimento e di saperla testare su campioni di dati.
- Dovrà essere in grado di generalizzare soluzioni per una classe di problemi con lo stile della programmazione strutturata.

Conoscenza e capacità di comprensione applicate:

- Dovrà essere in grado di individuare una strategia di soluzione che prevede la scomposizione del problema in sottoproblemi e di saper rappresentare sia la scomposizione sia gli algoritmi con adeguati linguaggi di descrizione.
- Dovrà dimostrare di saper implementare la soluzione proposta utilizzando il linguaggio imperativo di riferimento e di saperla testare su campioni di dati.

Autonomia di giudizio:

- Lo studente dovrà essere in grado di correggere e validare il corretto funzionamento dei programmi sviluppati.
- Dovrà essere in grado di discutere le soluzioni proposte chiarendo le scelte progettuali e implementative.

Abilità comunicative:

- Lo studente dovrà essere in grado di rendere il codice scritto comprensibile ad altri, mediante la sua descrizione generale e commenti specifici alle istruzioni e alle strutture di controllo utilizzate.
- Dovrà dimostrare di aver acquisito piena conoscenza dei concetti presentati a lezione nonché degli algoritmi fondamentali.

Capacità di apprendere:

- Lo studente dovrà essere in grado di trasformare autonomamente algoritmi descritti con flowchart in programmi in linguaggio C;
- Lo studente dovrà essere in grado di utilizzare le soluzioni alternative descritte nel testo di riferimento, se non descritte nel corso delle lezioni, come ad esempio le diverse modalità di dichiarazione delle variabili.

</td></tr>
<tr><td>Criteri di misurazione dell'apprendimento e di attribuzione del voto finale</td><td>

Voto	Descrittori
< 18 insufficiente	Conoscenze frammentarie e superficiali dei contenuti, errori nell'applicare i concetti, descrizione carente.
18 - 20	Conoscenze dei contenuti sufficienti ma generali, descrizione semplice, incertezze nell'applicazione di concetti teorici.
21 - 23	Conoscenze dei contenuti appropriate ma non approfondite, capacità di applicare i concetti teorici, capacità di presentare i contenuti in modo semplice.
24 - 25	Conoscenze dei contenuti appropriate ed ampie, discreta capacità di applicazione delle conoscenze, capacità di presentare i contenuti in modo articolato.
26 - 27	Conoscenze dei contenuti precise e complete, buona capacità di applicare le conoscenze, capacità di analisi, descrizione chiara e corretta.
28 - 29	Conoscenze dei contenuti ampie, complete ed approfondite, buona applicazione dei contenuti, buona capacità di analisi e di sintesi, descrizione sicura e corretta.
30 30 e lode	Conoscenze dei contenuti molto ampie, complete ed approfondite, capacità ben consolidata di applicare i contenuti, ottima capacità di analisi, di sintesi e di collegamenti interdisciplinari, padronanza di descrizione.

</td></tr>
<tr><td>Altro</td><td>Si suggerisce agli studenti di affidarsi esclusivamente alle informazioni/comunicazioni fornite sui siti ufficiali del Dipartimento di Informatica,</td></tr>
</table>

ovvero sui gruppi social solo se costituiti e amministrati esclusivamente dai docenti dei relativi insegnamenti:

- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea
- https://www.uniba.it/it/ricerca/dipartimenti/informatica
- https://elearning.uniba.it/

I programmi di tutti gli insegnamenti sono disponibili al seguente link:
- https://elearning.uniba.it/

Le informazioni che tutti gli studenti dovrebbero conoscere sono scritte nei regolamenti didattici dei Corsi di Studi disponibili nel sito:
- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea

Si suggerisce agli studenti di diffidare delle informazioni e dei materiali circolanti su siti o gruppi social non ufficiali, poiché spesso sono risultati non affidabili, non corretti o incompleti. Per ogni dubbio, chiedere un incontro al docente secondo le modalità previste per il ricevimento.

Il link per accedere al materiale del corso è il seguente https://elearning.uniba.it/course/view.php?id=2830
La chiave di iscrizione sarà fornita durante le lezioni.

NOTE (**da eliminare al termine della compilazione del programma in inglese**):

- In italiano la didattica è organizzata in didattica frontale, o lezione, e in esercitazione con presenza del docente (in aula o in laboratorio). In inglese chiamiamo "class" sia una lezione che una esercitazione col docente, chiamiamo "lecture" una lezione col docente e "practice session" una esercitazione in aula o in laboratorio.
- Gli appelli d'esame nell'anno accademico li chiamiamo "official dates of the exam in the academic year"
- La prenotazione all'esame su ESSE3 la chiamiamo "registration to the exam in ESSE3"
- La prova di esonero a metà semestre la chiamiamo "middle term partial exam"

Main information on the course	
Course name	Programming (Track E-M)
Degree	Informatica
Academic year	2025/26
European Credit Transfer and Accumulation System (ECTS), in Italian Crediti Formativi Universitari (CFU)	9 CFU (each CFU corresponds to 25 hours (h) of student's time); CFU are of type T1, T2 or T3 T1 = 8 h lecture + 17 h individual study T2 = 15 h practice + 10 h individual study T3 = 25 h individual study
Settore Scientifico Disciplinare	INF/01
Course language	Italian
Course year	First
Course period	First Semester - exact dates can be found in the didactic regulations
Course attendance requirement	None, but it is highly recommended to attend classes
Website of the Degree	https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea

Teacher(s)	
Name and Surname	Veronica Rossano
email	Veronica.rossano@uniba.it
phone	+39 080 544 2477
office	Department of Computer Science, Via Orabona 4, 70125, Bari. Room n.772, 7^ floor.
e-learning platform	UNIBA e-learning platform - https://elearning.uniba.it/course/view.php?id=2830
Teacher's homepage	https://www.uniba.it/it/docenti/rossano-veronica
Office hours	Every day by appointment by e-mail or on MsTeams.

Syllabus	
Course goals	The course aims to introduce problem solving methodology and the basic elements of structured imperative programming to formulate algorithmic solutions to problems of varying complexity. In particular, the student will acquire the ability to apply problem solving techniques, to identify solution strategies of an algorithmic nature and to use the C programming language to implement them.
Prerequisites/requirements	No computer skills of any kind are required to attend the Computer Science degree course, but a good preparation in basic secondary school subjects is indispensable, particularly mathematical, logical and reasoning skills.

Course program

Module	Topics	Estimated Hours
1	Course presentation, content, exam methods, exemptions, attendance, class schedules, exercise methods, etc. Introduction to Programming: from problem to program.	3
2	Algorithm as an abstraction. Properties of algorithms.	3
3	Problem Solving and the analysis phase. Problem analysis. Problem formulation and its specifications. Problem data. Expected results. Errors and edge cases. Ambiguities, value limits, and data controls. Examples and exercises.	6
4	Problem Solving and the design phase. Decomposing the problem into subproblems. Techniques: top-down, bottom-up, hybrid. Building the solution method. Algorithm representation through representation languages: structured flowcharts, linear language, decomposition tree, Nassi-Schneidermann diagrams. Bohm-Jacopini theorem: statement. Examples and exercises.	15
5	Simple and complex problems. Sequential, selective, iterative, and recursive decomposition. Examples and exercises.	3
6	The program. Variables. Types of instructions. Declarative instructions. Input-output instructions. Assignment instruction. Arithmetic expressions. Logical expressions. Constants. Control structures. Examples and exercises.	3
7	Elementary algorithms: counting, summation, and average of a set of numbers, factorial, conversion from characters to numbers in base 10, prime number, greatest common divisor, Fibonacci series, swapping.	3
8	Data types. Type declaration. Taxonomy of data types. Standard types. User-defined simple types: enumerative, subrange.	3
9	Data types. Structured types. Structured data and Array type. Data structure and Record type. Internal representation of arrays and records. Array indices. Examples and exercises.	7
10	Exercise and preparation for the self-assessment test: problem analysis, subproblem decomposition, algorithmic design of the solution. Cooperative participatory activity conducted with the assistance of the instructor.	3
11	Self-assessment exercise: solving a problem according to the phases of problem-solving except coding.	4
12	Self-correction of the self-assessment test assisted by the instructor.	2
13	Modular programming. Subprograms. Utility of subprograms. Subprogram as a functional abstraction. Resulting structure of a program. Subprogram calls. Nesting. Examples.	8
14	Communication of subprograms with the external environment and the calling environment. Subprogram view. Shadowing effect. Global, local, and non-local variables. Visibility rules. Scope and duration of variables. Examples and exercises.	3

15	Subprograms. Static and dynamic allocation. Side effect. Contour model. Examples.	3
16	Subprograms. Actual and formal parameters. Parameter passing methods: by value, by reference, and by name. Evaluation and comparison of passing methods. Examples and exercises.	4
17	Subprograms. Procedures. Procedure header and call. Examples and exercises.	3
18	Subprograms. Functions. Function header and call. Procedures vs functions. Examples and exercises.	3
19	Subprograms. Subprograms as parameters. Subprogram activation. Activation records and the stack. Static and dynamic linking. Execution management. Examples and exercises.	4
20	Recursion. Recursive functions. Execution management with recursive functions. Examples and exercises.	3
21	Fundamental algorithms: Algorithms on arrays: printing histograms using arrays, removing duplicates from sorted arrays. Algorithms on matrices. Linear search. Binary search. Array merging.	3
22	Sorting algorithms: selection sort, exchange sort, insertion sort. Recursive algorithms: factorial.	3
23	Structure of C programs. Data types: simple predefined – int, float, double, and char – explicit enumeration of values. Type definition.	3
24	Structured types in C: array constructor, struct constructor, and pointer constructor. Type compatibility.	4
25	Control structures in C. Selection instructions: If, Switch; Iterative instructions: while, do-while, for.	3
26	Functions and procedures in C: definition, call, prototype, parameter passing.	3
27	Variable visibility scope in C. Arrays as parameters. Structures as parameters.	4
28	Side effects. Predefined procedures and functions. Standard library. Pointers. Files.	5

Books of reference	**Primary text:** P. Deitel and H. Deitel, C Programming Language – Fundamentals and Programming Techniques, 8th Edition - Pearson 2016 - ISBN: 9788891901651 (subsequent and previous editions from the 4th onwards are also acceptable). **Optional supplementary text:** W. B. Kernighan, D.M. Ritchie. The C Programming Language. Principles of Programming and Reference Manual. Students can borrow the texts from the library. It is advisable to check availability through the University Library System https://opac.uniba.it/easyweb/w8018/index.php? and contact the library to arrange the loan.
Notes to the books	During the lectures, the instructor will illustrate concepts using slides that summarize the course content. The slides will be made available at the end of each lesson on the ADA platform of the department (see 'virtual office' above). Additionally, all exam simulations already carried out are available on the ADA platform.

Organization of the didactic activities

Hours				
Total	Lectures	Practice sessions	Project work	Individual study
300 hours	72 Hours	45 hours	hours	183 hours
CFU/ETCS				
12 CFU	9 CFU	3 CFU	CFU	

Teaching methods

The course is organized into lectures with slides, guided exercises conducted in the classroom, and exercises carried out independently at home. The guided exercises will be conducted in the classroom using the Bring Your Own Device (BYOD) approach. The exercises conducted in the classroom and at home must be completed individually by each student and submitted on the ADA platform according to the guidelines and deadlines communicated during the lectures. The exercises submitted will be useful for the instructor to verify participation in the exercises and understanding of the topics covered in class. Non-attending students are not required to submit the exercises.

Expected learning outcomes

Knowledge and understanding

- Understand the basic elements of programming.
- Understand problem-solving techniques.
- Understand the process of translating an algorithm into a program using a programming language.

Applying knowledge and understanding

- Apply problem-solving techniques to identify and formulate algorithmic solutions.
- Use the C programming language to implement algorithmic solutions.
- Develop programs that meet specified requirements.

Other skills

Making judgements

- Evaluate different problem-solving approaches.
- Assess the effectiveness of algorithmic solutions.
- Make informed decisions about the most appropriate programming constructs to use.

Communication

- Communicate effectively about problem-solving approaches and solutions.
- Use appropriate terminology to describe programming concepts and structures.

Learning skills

- Develop the ability to learn independently.
- Acquire skills to keep up-to-date with advancements in programming and problem-solving methodologies.

Assessment	
Assessment methods	Formative Assessment (useful to motivate students attending the class) • Exercises to be carried out during the course (not mandatory). • Participation in the exercises and their submission, according to the methods and deadlines communicated during the lectures, contribute to awarding a bonus (from 0 to 2 points in proportion to the number and quality of the submitted exercises). This bonus can be used only until the end of the first session (June/July exams). • Exercises submitted beyond the communicated deadlines or in ways different from those communicated will not be considered. Not completing the exercises during the course does not prejudice the achievement of the maximum grade in the exam. **Summative Assessment (Exam):** • Written/Laboratory test: The test takes place in the computer labs and consists of solving a complex problem. The test lasts approximately 3 hours and requires applying all phases of problem-solving, from analysis to design and finally coding in the C language. The written test is preparatory to the oral exam and is considered passed with a minimum score of 16/30. • Oral test: This involves reviewing the written test and discussing any mistakes made. During the oral test, two questions related to the theoretical part covered during the course will be asked. The middle term partial exam, in written form and consisting of close and open-ended questions and C programming exercises, is held after a consistent number of topics. It will be graded out of thirty, and is considered passed if the score is greater than 18. Passing the exemption test exempts the student from the oral test. The grades from this middle term partial exam can be used exclusively until **September**. The student can, if preferred, waive the grade of the exemption test and take the oral test. The final grade achieved is calculated as the average of the grades obtained in the two tests (laboratory test and oral test/exemption test) and is communicated via the Esse3 platform.
Evaluation criteria	

	Knowledge and Understanding: • The student should be able to analyze problems by formulating additional hypotheses, identifying the necessary and sufficient data for the solution, and providing a description of it. • They should be able to identify a solution strategy that involves breaking the problem down into sub-problems and be able to represent both the decomposition and the algorithms with appropriate description languages. • They should demonstrate the ability to implement the proposed solution using the reference imperative language and test it on data samples. • They should be able to generalize solutions for a class of problems using the style of structured programming. **Applied Knowledge and Understanding:** • They should be able to identify a solution strategy that involves breaking the problem down into sub-problems and be able to represent both the decomposition and the algorithms with appropriate description languages. • They should demonstrate the ability to implement the proposed solution using the reference imperative language and test it on data samples. **Judgment Autonomy:** • The student should be able to correct and validate the correct functioning of the developed programs. • They should be able to discuss the proposed solutions, clarifying the design and implementation choices. **Communication Skills:** • The student should be able to make the written code understandable to others through its general description and specific comments on the instructions and control structures used. • They should demonstrate having fully acquired knowledge of the concepts presented in class as well as the fundamental algorithms. **Learning Skills:** • The student should be able to independently transform algorithms described with flowcharts into programs in the C language. • The student should be able to use alternative solutions described in the reference text, if not covered during the lessons, such as different ways of declaring variables.
Measurements and final grade	

Voto	Descrittori
< 18 insufficiente	Fragmentary and superficial knowledge of the contents, errors in applying the concepts, deficient description.
18 - 20	Sufficient but general content knowledge, simple description, uncertainties in the application of theoretical concepts.
21 - 23	Appropriate but not in-depth knowledge of content, ability to apply theoretical concepts, ability to present content in a simple way.
24 - 25	Appropriate and extensive knowledge of the contents, good ability to apply knowledge, ability to present the contents in an articulated way.
26 - 27	Precise and complete content knowledge, good ability to apply knowledge, analytical skills, clear and correct description.
28 - 29	Wide, complete and in-depth knowledge of the contents, good application of the contents, good capacity for analysis and synthesis, safe and correct description.
30 30 e lode	Very broad, complete and in-depth knowledge of the contents, well-established ability to apply the contents, excellent capacity for analysis, synthesis and interdisciplinary connections, mastery of description.

Further information

Students are advised to rely exclusively on the information/communications provided on the official websites of the Computer Science Department, or on social groups only if set up and administered exclusively by the teachers of the related courses:

- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-dilaurea/corsi-di-laurea
- https://www.uniba.it/it/ricerca/dipartimenti/informatica
- https://elearning.di.uniba.it/

Course schedules are available here:

- https://programmi.di.uniba.it/

The information that all students should know is written in the Teaching regulations and study posters available on the site:

- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-dilaurea/corsi-di-laurea

Students are advised to be wary of information circulating on unofficial sites or social groups, as they are often found to be unreliable, incorrect or incomplete.

Link to the course on the ADA department e-learning platform:
https://elearning.di.uniba.it/