Principali informazioni sull'insegnamento	
Denominazione dell'insegnamento	**Programmazione (cognomi A-L)**
Corso di studio	Informatica e Tecnologie per la Produzione del Software
Anno Accademico	2024/25
Crediti formativi universitari (CFU) / European Credit Transfer and Accumulation System (ECTS)	12 CFU
Settore Scientifico Disciplinare	ING-INF/05-Sistemi di Elaborazione dell'Informazione
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/corsi/cdl-informatica-tecnologie-produzione-software

Docente	
Nome e cognome	Mario Alessandro Bochicchio
Indirizzo mail	mario.bochicchio@uniba.it
Telefono	+39 080 544 3265
Sede	Dipartimento di Informatica, Via Orabona 4, 70125, Bari. Stanza n.525, 5^ piano.
Sede virtuale	Piattaforma ADA - https://elearning.di.uniba.it/
Sito web del docente	https://persone.ict.uniba.it/rubrica/mario.bochicchio/
Ricevimento (giorni, orari e modalità, es. su appuntamento)	Giovedì 15:00-17:00 e anche su appuntamento via email o MS-Teams

Syllabus	
Obiettivi formativi	Il corso si propone di introdurre gli elementi base della programmazione imperativa strutturata per formulare soluzioni algoritmiche a problemi di complessità limitata.

	In particolare, lo studente acquisirà la capacità di usare il linguaggio di programmazione C come strumento per modellare problemi e formalizzarne le soluzioni
Prerequisiti	Buona comprensione della lingua inglese. Lettura individuale da parte dello studente del primo capitolo del testo di riferimento indicato nel seguito.
Contenuti di insegnamento (Programma)	

Mod	Argomenti	Ore
1	Presentazione del corso, contenuti, modalità d'esame, esoneri, frequenza alle lezioni, orari, modalità di esercitazione in aula, ecc. Risposte alle domande degli studenti. Introduzione alla programmazione Un semplice programma C: visualizzare una riga di testo. Un altro semplice programma C: sommare due interi.	3
2	Uso del compilatore ambiente di sviluppo VS Code cenni al debugging esercitazioni e approfondimenti degli argomenti: Un semplice programma C: visualizzare una riga di testo.	3
3	Nozioni sulla memoria L'aritmetica del C Prendere delle decisioni: gli operatori di uguaglianza e relazionali. Gli algoritmi Lo pseudocodice. Le strutture di controllo Il comando di selezione if	3
4	Il comando di selezione if else. Il comando di iterazione while. Formulazione degli algoritmi: studio di un caso: l'iterazione controllata da un contatore. Formulazione degli algoritmi con processo top down per raffinamenti successivi: studio di un caso: iterazione controllata da un valore sentinella. Conversione di tipo, precisione.	4
5	Strutture di controllo nidificate esercitazione con uso di pseudocodice Formulazione degli algoritmi con processo top down per raffinamenti successivi: studio di un caso: strutture di controllo nidificate	3
6	Gli operatori di incremento e di decremento Gli elementi dell'iterazione Iterazione controllata da un contatore il comando di iterazione FOR note e osservazioni esempi di utilizzo del comando FOR Il comando di selezione multipla switch introduzione al comando break, studio di un caso. Il comando di iterazione do...while Le istruzioni break e continue	4
7	Gli operatori logici Operatori di uguaglianza (==) e di assegnamento (=); Riassunto della programmazione strutturata. I moduli di programma in C	3
8	Le funzioni della libreria matematica Le funzioni Le definizioni di funzione I prototipi di funzione Lo stack delle chiamate di funzione e i record di attivazione	3

	I file di intestazione Invocare le funzioni: chiamata per valore e per riferimento Generazione di numeri casuali	
9	Esercitazione/autovalutazione: sviluppo flowchart, attività svolta con assistenza del docente	4
10	Esempio: un gioco d'azzardo Le classi di memoria Le regole di visibilità	4
11	La ricorsione	2
12	I vettori La dichiarazione dei vettori Esempi sui vettori	3
13	Esercitazione e studio individuale di materiale video prodotto dal Docente, a supporto delle lezioni teoriche. Il video descrive lo sviluppo di un flowchart prendendo spunto dall'esercizio del cap 4.38 del testo 'Il linguaggio C' di Deitel&Deitel. Il metodo di progettazione rispetta i principi della programmazione strutturata e utilizza strutture di controllo di base del linguaggio di programmazione C	4
14	Ulteriori esempi sui vettori	3
15	Vettori statici ed automatici Passare i vettori alle funzioni, passaggio per riferimento (indirizzo) e per valore, qualificatore 'const' Algoritmi di ordinamento: bubble sort	4
16	Studio di un caso: calcolare la media, mediana e la moda usando i vettori La ricerca nei vettori (lineare), algoritmo e programma per la ricerca lineare in un vettore	4
17	La ricerca nei vettori (binaria) Vettori multidimensionali.	4
18	Manipolazione delle matrici esercitazione sulle matrici bidimensionali (fig.6.22 del testo)	3
19	Esercitazione e studio individuale di materiale video prodotto dal Docente, a supporto delle lezioni teoriche. Il video descrive lo sviluppo di un flowchart prendendo spunto dall'esercizio 'dado a 20 facce' del testo 'Il linguaggio C' di Deitel&Deitel. Il metodo di progettazione rispetta i principi della programmazione strutturata e utilizza strutture di controllo di base del linguaggio di programmazione C	3
20	Esercitazione: Individuazione dei numeri primi. Crivello di Eratostene.	3
21	Esercitazione e studio individuale di materiale video prodotto dal docente, a supporto delle lezioni teoriche. Il video descrive il passaggio da un algoritmo descritto con flowchart al codice in Linguaggio C, prendendo spunto dall'esercizio 'dado a 20 facce' del testo 'Il linguaggio C' di Deitel&Deitel. Il metodo di progettazione rispetta i principi della programmazione strutturata e utilizza strutture di controllo di base del linguaggio di programmazione C	3
22	Esercitazione e studio individuale di materiale video prodotto dal Docente, a supporto delle lezioni teoriche. Il video descrive lo sviluppo di un flowchart prendendo spunto dall'esercizio 'Craps - un gioco d'azzardo' del testo 'Il linguaggio C' di Deitel&Deitel. Il metodo di progettazione rispetta i principi della programmazione strutturata e utilizza strutture di controllo di base del linguaggio di programmazione C	3
23	La gerarchia dei dati I file e gli stream Creare un file ad accesso sequenziale	4

	Leggere i dati da un file ad accesso sequenziale	
24	Leggere i dati da un file ad accesso sequenziale (approfondimenti) Programma per l'interrogazione del credito.	3
25	I file ad accesso casuale Creare un file ad accesso casuale	4
26	Scrivere i dati in modo casuale in un file ad accesso casuale Leggere i dati in modo casuale da un file ad accesso casuale	3
27	Studio di un caso: un programma per l'elaborazione delle transazioni	3
28	Approfondimenti sulla ricorsione ed esercizi: fattoriale ricorsivo, Fibonacci, Ricerca binaria ricorsiva.	3
29	Puntatori: introduzione dichiarazione e inizializzazione operatore di indirizzo (&) e operatore di dereferenziazione (*) passaggio per valore e per riferimento puntatori come parametri di funzioni Esercizi: dichiarazione e stampa dei valori contenuti in variabili e puntatori, elevare al cubo una variabile (chiamata per valore e chiamata per riferimento), trovare il massimo e il minimo di un vettore usando il passaggio per riferimento.	3
30	Puntatori: puntatori const, operatori sui puntatori, esercizio bubble sort con puntatori, esercizio bubble sort in ordine crescente e decrescente usando puntatori a funzione, Array multidimensionali ed esercizio mescola carte	4
31	Correzioni collettive esercitazione svolta. Attività didattica di preparazione all'esonero.	4
32	simulazione autonoma in preparazione del 1^ esonero, assistita dal docente	3
33	Auto correzione esonero assistita dal docente	4
34	esercitazione autonoma, assistita dal docente; Correzione collettiva esercitazione svolta. Attività didattica di preparazione all'esonero.	3
35	Esercitazione e Simulazione 2^ esonero	3

Testi di riferimento

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)

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

Il testo di riferimento contiene tutti gli argomenti del corso, pertanto si consiglia di studiare dal testo e di svolgere in autonomia e costantemente tutti gli esercizi inseriti alla fine di ogni capitolo trattato a lezione.
Sulla piattaforma ADA del dipartimento (v. sopra 'sede virtuale') sono disponibili:

- Il materiale di supporto utilizzato a lezione;
- alcuni esempi di esercizi e tracce svolte.

Organizzazione della didattica

Ore

Totali	Didattica frontale	Laboratorio/esercitazione	Progetto	Studio individuale
300 ore	72 ore	45 ore	0 ore	183 ore

CFU/ETCS				
12 CFU	9 CFU	3 CFU	0 CFU	

Metodi didattici

	Lezioni frontali, esercitazioni ed attività autonome e di gruppo in aula e a casa (come dettagliato nel programma). Gli studenti non frequentanti possono lavorare singolarmente prendendo accordi con il docente.

Risultati di apprendimento previsti

Conoscenza e capacità di comprensione	● 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;
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 (es. Xcode, Visual Studio) 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;

	● 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 video prodotto e proposto dal docente; ● Capacità di completare autonomamente il percorso formativo previsto dal testo di riferimento, oltre i contenuti previsti dal programma dell'insegnamento.

Valutazione	
Modalità di verifica dell'apprendimento	Una prova di valutazione intermedia, con valore esonerante, si tiene in prossimità della settimana di interruzione delle lezioni, normalmente collocata intorno alla metà di novembre. La prova consiste nella soluzione di sei di natura teorico-pratica, che includono la definizione di semplici algoritmi e lo sviluppo del relativo codice in C, analogamente a quanto spiegato nel corso delle lezioni. La valutazione sarà "insufficiente", "sufficiente" o "buono". La prova finale consiste: - nella presentazione al docente, e contestuale debugging, di un esercizio di programmazione tra 25 scelti dallo studente (almeno 2 per ognuno dei capitoli dal 2 al 11 del libro di testo); - nella risposta a domande teorico-pratiche indicate dal docente in sede di esame e relative agli argomenti presentati discussi a lezione. Gli studenti che scelgono di non partecipare alla prova di valutazione intermedia, o che non la superano con "sufficiente" o "buono", in occasione della prova finale dovranno svolgere due esercizi scritti di programmazione. La valutazione sarà "insufficiente", "sufficiente" o "buono". Il voto finale: - per coloro che hanno ottenuto "sufficiente" alla prova intermedia o agli esercizi scritti di programmazione, varrà al massimo 26/30; - per coloro che hanno ottenuto "buono" alla prova intermedia o agli esercizi scritti di programmazione, potrà raggiungere 30/30; Il giudizio dell'esonero può essere utilizzato esclusivamente nella prima sessione di esami (gennaio/febbraio). Materiali permessi per sostenere la prova intermedia e la prova finale: testo di riferimento in formato cartaceo e *cheat sheets* indicati dal docente. Incentivi alla frequenza: la partecipazione attiva alle lezioni, l'aver proposto soluzioni, risolto casi proposti dal docente e sviluppato attività attinenti al corso e concordate con il docente possono contribuire al conseguimento della lode.
Criteri di valutazione	

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

● **Conoscenza e capacità di comprensione:**
○ Lo studente dovrà essere in grado di analizzare e risolvere semplici problemi e di generalizzare soluzioni per una classe di problemi con lo stile della programmazione strutturata.

● **Conoscenza e capacità di comprensione applicate:**
○ Lo studente dovrà essere in grado di codificare le soluzioni ideate descrivendole in pseudocodice, in flowchart strutturati e nel linguaggio di programmazione C;
○ Lo studente dovrà essere in grado di utilizzare un ambiente di sviluppo a sua scelta e dimostrare di conoscere il linguaggio C;

● **Autonomia di giudizio:**
○ Lo studente dovrà essere in grado di correggere e validare il corretto funzionamento dei programmi sviluppati.

● **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.

● **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, ovvero sui gruppi social solo se costituiti e amministrati esclusivamente dai docenti dei relativi insegnamenti:</td>
</tr>
</table>

- 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.

Link al corso sulla piattaforma e-learning del dipartimento ADA:
https://elearning.di.uniba.it/

Link al corso sulla piattaforma e-learning del dipartimento MS Teams:
da richiedere scrivendo al docente: mario.bochicchio@uniba.it.

UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO | DIPARTIMENTO DI INFORMATICA

Main information on the course

Course name	**Programmazione (family names A-L)**
Degree	Informatica e Tecnologie per la Produzione del Software
Academic year	2024/25
European Credit Transfer and Accumulation System (ECTS), in Italian Crediti Formativi Universitari (CFU)	12 CFU
Settore Scientifico Disciplinare	ING-INF/05-Sistemi di Elaborazione dell'Informazione
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/corsi/cdl-informatica-tecnologie-produzione-software

Teacher(s)

Name and Surname	Mario Alessandro Bochicchio
email	mario.bochicchio@uniba.it
phone	+39 080 544 3265
office	Dipartimento di Informatica, Via Orabona 4, 70125, Bari. Stanza n.525, 5^ piano.
e-learning platform	ADA platform - https://elearning.di.uniba.it/
Teacher's homepage	https://persone.ict.uniba.it/rubrica/mario.bochicchio/
Office hours	Thursday 15:00-17:00, also by appointment via email or MS-Teams

Syllabus

Course goals

The course aims to introduce the basic elements of structured imperative programming to formulate algorithmic solutions to problems of limited complexity. In particular, the student will acquire the ability to use the C programming language as a tool for modeling problems and formalizing their solutions

Prerequisites/requirements

Good understanding of the English language. Individual reading by the student of the first chapter of the reference text given below.

Course program

Mod	Argomenti	Ore
1	Course presentation, content, exam modes, waivers, class attendance, schedule, classroom practicè modes, etc. Answers to students' questions. Introduction to programming A simple C program: display a line of text. Another simple C program: sum two integers.	3
2	Using the compiler VS Code development environment hints at debugging tutorials and in-depth discussion of topics: A simple C program: displaying a line of text.	3
3	Notions of memory	3

			The arithmetic of C Decision making: equality and relational operators. The algorithms Pseudocode. The control structures. The if selection command	
		4	The if else selection command. The while iteration command. Formulation of algorithms: study of a case: iteration controlled by a counter. Formulation of algorithms with top down process for successive refinements: study of a case: iteration controlled by a sentinel value. Type conversion, precision.	4
		5	Nested control structures exercise with use of pseudocode Algorithm formulation with top down process for successive refinements: case study: nested control structures	3
		6	The increment and decrement operators The elements of iteration Iteration controlled by a counter the FOR iteration command notes and observations examples of using the FOR command The switch multiple selection command introduction to the break command, case study. The do...while iteration command The break and continue statements	4
		7	The logical operators Equality (==) and assignment (=) operators; Summary of structured programming. Program modules in C	3
		8	The functions of the mathematical library The functions The definitions of functions The function prototypes The function call stack and activation records The header files Invoking functions: calling by value and by reference Generating random numbers	3
		9	Exercise/self-evaluation: flowchart development, activity carried out with teacher assistance	4
		10	Example: a game of chance The memory classes The visibility rules	4
		11	Recursion	2
		12	Vectors The declaration of vectors Examples on vectors	3
		13	Exercise and individual study of video material produced by the Lecturer to support the theoretical lectures. The video describes the development of a flowchart taking its cue from the exercise in ch. 4.38 of the text 'The C Language' by Deitel&Deitel. The design method respects the principles of structured programming and uses basic control structures of the C programming language	4
		14	More examples on vectors	3
		15	Static and automatic vectors Passing vectors to functions, passing by reference (address) and by value, 'const' qualifier Sorting algorithms: bubble sort	4

		16	Case study: calculating mean, median and mode using vectors The search in vectors (linear), algorithm and program for linear search in a vector	4
		17	The search in vectors (binary) Multidimensional vectors.	4
		18	Handling of matrices Exercise on two-dimensional matrices (fig.6.22 in the text)	3
		19	Exercise and individual study of video material produced by the Lecturer to support the theoretical lectures. The video describes the development of a flowchart taking a cue from the '20-sided die' exercise from the text 'The C Language' by Deitel&Deitel. The design method respects the principles of structured programming and uses basic control structures of the C programming language.	3
		20	Exercise: Identification of prime numbers. Eratosthenes' sieve.	3
		21	Exercise and individual study of material produced by the lecturer to support the theoretical lectures. The material describes the transition from an algorithm described with flowchart to code in C Language, taking its cue from the '20-sided die' exercise in the text 'The C Language' by Deitel&Deitel. The design method respects the principles of structured programming and uses basic control structures of the C programming language	3
		22	Exercise and individual study of video material produced by the Lecturer to support the theoretical lectures. The video describes the development of a flowchart taking a cue from the exercise 'Craps - a game of chance' from the text 'The C Language' by Deitel&Deitel. The design method respects the principles of structured programming and uses basic control structures of the C programming language	3
		23	The data hierarchy Files and streams Creating a sequentially accessed file Reading data from a sequentially accessed file	4
		24	Reading data from a sequential access file (more details) Credit query program.	3
		25	Random access files Creating a random access file	4
		26	Write data randomly to a random access file Read data randomly from a random access file	3
		27	Case study: a transaction processing program	3
		28	Insights into recursion and exercises: recursive factorial, Fibonacci, Recursive Binary Search.	3
		29	Pointers: introduction declaration and initialization address operator (&) and dereference operator (*) passing by value and by reference pointers as parameters of functions Exercises: declaring and printing values contained in variables and pointers, cubing a variable (call by value and call by reference), finding the maximum and minimum of a vector using passing by reference.	3
		30	Pointers: const pointers, operators on pointers, bubble sort exercise with pointers, bubble sort exercise in ascending and descending order using function pointers, Multidimensional arrays and exercise shuffle cards	4

31	Collective corrections exercise carried out. Teaching activities in preparation for middle term partial exam.	4
32	Autonomous simulation in preparation for the middle term partial exam, assisted by the teacher	3
33	Teacher-assisted middle term partial exam self-correction	4
34	Independent, instructor-assisted exercise; Collective correction exercise. Teaching activities in preparation for middle term partial exam.	3
35	Exercise and Simulation final exam.	3

Books of reference

P. Deitel e H. Deitel
Il linguaggio C – Fondamenti e tecniche di programmazione
8^edizione - Pearson 2016 - ISBN: 9788891901651

Students who wish to do so can obtain texts on loan from the library. It may be convenient to check their availability through the University Library System https://opac.uniba.it/easyweb/w8018/index.php? and contact the library to arrange borrowing.

Notes to the books

The reference text contains all the topics of the course, so it is advisable to study from the text and to do independently and consistently all the exercises included at the end of each chapter covered in class.
Available on the department's ADA platform (see 'virtual seat' above) are:
- The supporting materials used in class;
- Some examples of exercises and traces carried out.

Organization of the didactic activities

Hours

Total	Lectures	Practice sessions	Project work	Individual study
300 hours	72 hours	45 hours	0 hours	183 hours
CFU/ETCS				
12 CFU	9 CFU	3 CFU	0 CFU	

Teaching methods

Lectures, tutorials and independent and group activities in the classroom and at home (as detailed in the syllabus). Non-attending students may work individually by making arrangements with the lecturer.

Expected learning outcomes

Knowledge and understanding

- Acquire knowledge that enables the student to understand how to indicate to an electronic processor (automatic machine of universal use, hardware) the solution of a problem or class of problems, which the processor can solve, using an appropriate method and language, by creating an appropriate program (software) executable by the processor.
- Acquire the ability to reason and identify a solution to a problem (algorithm) according to the paradigm of structured imperative programming;

Applying knowledge and understanding	● Understand the use of an unconventional design language (e.g., pseudocode) and the use of a graphical representation (e.g., flow chart) to describe an algorithm with a simple formalism; ● Understand the vocabulary, syntax and semantics of the C programming language; ● Acquire the ability to write a structured program in C language; ● Acquire the ability to identify test cases for the domain referenced by the created program; ● Acquire the ability to use a development environment (e.g., Xcode, Visual Studio) to transform the source program (in C) into an executable program and run it.
Other skills	*Making judgements* ● Acquire the ability to verify that the algorithm identified meets the specifications of a problem; ● Acquire the ability to verify that the results obtained after program execution are as expected. *Communication* ● Learn to comment on the code produced in order to make it understandable and easily editable by other professionals, with the goal of team development. *Learning skills* ● Ability to deepen concepts through independent study of video material produced and proposed by the lecturer; ● Ability to independently complete the training provided by the reference text, beyond the content provided in the teaching syllabus.

Assessment	
Assessment methods	A middle term partial exam is held near the week of class break, normally placed around mid-November. The test consists of solving six of a theoretical-practical nature, including the definition of simple algorithms and the development of related code in C, similar to what was explained in the course of lectures. The grade will be "insufficient," "sufficient," or "good." The final exam consists of: - in the presentation to the lecturer, and contextual debugging, of one programming exercise from 25 chosen by the student (at least 2 from each of chapters 2 through 11 of the textbook); - in answering theoretical-practical questions indicated by the lecturer in the exam and related to the presented topics discussed in class. Students who choose not to participate in the midterm assessment test, or who fail it with "sufficient" or "good," will be required to do two written programming exercises on the final exam. The grade will be "insufficient," "sufficient" or "good."

	The final grade: - for those who obtained "sufficient" on the midterm or written programming exercises, will be worth a maximum of 26/30; - for those who obtained "good" at the midterm test or written programming exercises, it may reach 30/30; The waiver judgment can only be used in the first exam session (January/February).
Evaluation criteria	● Knowledge and understanding skills: o The student should be able to analyze and solve simple problems and generalize solutions for a class of problems with the style of structured programming. ● Applied knowledge and understanding skills: o The student should be able to code the devised solutions by describing them in pseudocode, structured flowcharts, and the C programming language; o The student should be able to use a development environment of his/her choice and demonstrate knowledge of the C language; ● Autonomy of judgment: o The student should be able to correct and validate the correct operation of developed programs. ● Communication skills: o The student should be able to make the written code understandable to others through its general description and specific comments to the instructions and control structures used. ● Learning skills: o The student shall be able to independently transform algorithms described with flowcharts into C language programs; o The student should be able to use the alternative solutions described in the reference text, if not described in the lectures, such as different ways of declaring variables.
Measurements and final grade	

Voto	Descrittori
< 18 insufficient	Fragmentary and superficial knowledge of content, errors in applying concepts, poor description.
18 - 20	Sufficient but generic content knowledge, simple description, uncertainties in applying theoretical concepts.
21 - 23	Appropriate but not extensive content knowledge, ability to apply theoretical concepts, ability to present content in a simple way.
24 - 25	Appropriate and extensive content knowledge, fair ability to apply knowledge, ability to present content in an articulate manner.
26 - 27	Content knowledge accurate and complete, good ability to apply knowledge, ability to analyze, clear and correct description.
28 - 29	Broad, complete and thorough content knowledge, good application of content, good ability to analyze and summarize, confident and correct description.
30 30 e lode	Very broad, complete and thorough content knowledge, well-established ability to apply content, excellent ability to analyze, synthesize and make interdisciplinary connections, mastery of description.

Further information	It is suggested that students rely exclusively on the information/communication provided on the official websites of the Department of Computer Science, or on

social groups only if they are established and administered exclusively by the faculty members of the relevant subjects:

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

Schedules of the teachings are available here:
- https://programmi.di.uniba.it/

Information that all students should know is written in the Teaching Regulations and Study Manifestos available on the website:
- https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea

Students are suggested to be wary of information and materials circulating on unofficial sites or social groups, as they are often found to be unreliable, incorrect or incomplete. If you have any doubts, ask for a meeting with the lecturer in accordance with the reception arrangements.

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

Link to the course on the e-learning platform of the MS Teams department:
To be requested by writing to the lecturer: mario.bochicchio@uniba.it.