

<b>Principali informazioni sull'insegnamento</b>										
Denominazione dell'insegnamento	<b>Programmazione</b>									
Corso di studio	Informatica e Comunicazione Digitale (Taranto)									
Anno Accademico	2024/25									
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	La frequenza è fortemente raccomandata									
Sito web del corso di studio	<a href="https://www.uniba.it/it/corsi/cdl-informatica-comunicazione-digitale-taranto/corso-di-laurea-triennale-in-informatica-e-comunicazione-digitale/">https://www.uniba.it/it/corsi/cdl-informatica-comunicazione-digitale-taranto/corso-di-laurea-triennale-in-informatica-e-comunicazione-digitale/</a>									
<b>Docente/i</b>										
Nome e cognome	Pasquale ARDIMENTO									
Indirizzo mail	pasquale.ardimento@uniba.it									
Telefono										
Sede	Dipartimento di Informatica, Via Orabona 4, 70125, Bari. Stanza n.569, V piano.									
Sede virtuale	Piattaforma ADA - <a href="https://elearning.di.uniba.it/">https://elearning.di.uniba.it/</a>									
Sito web del docente	<a href="https://www.uniba.it/it/docenti/ardimento-pasquale/">https://www.uniba.it/it/docenti/ardimento-pasquale/</a>									
Ricevimento	Martedì 14:30 – 15:30, o in altro giorno previo appuntamento per e-mail									
<b>Syllabus</b>										
<b>Obiettivi formativi</b>	<p>Il corso si propone di</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• introdurre i principi e i costrutti del paradigma Object-Oriented consentendo allo studente di acquisire le competenze e gli elementi base della programmazione imperativa strutturata per formulare soluzioni algoritmiche a problemi di complessità semplice e di usare il linguaggio Java per implementarle.</li> </ul>									
<b>Prerequisiti</b>	Non è richiesto alcun prerequisito in quanto trattasi di un insegnamento del primo semestre del primo anno.									
<b>Contenuti di insegnamento (Programma)</b>	<table border="1"> <thead> <tr> <th>Mod</th> <th>Argomenti (l'indicazione oraria si intende stimata)</th> <th>Ore</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Presentazione del corso, contenuti, modalità d'esame, frequenza alle lezioni, orari, modalità di esercitazione in aula, ecc. Introduzione alla Programmazione: dal problema al programma</td> <td>2</td> </tr> <tr> <td>2</td> <td>L'algoritmo come astrazione. Le proprietà dell'algoritmo</td> <td>2</td> </tr> </tbody> </table>	Mod	Argomenti (l'indicazione oraria si intende stimata)	Ore	1	Presentazione del corso, contenuti, modalità d'esame, frequenza alle lezioni, orari, modalità di esercitazione in aula, ecc. Introduzione alla Programmazione: dal problema al programma	2	2	L'algoritmo come astrazione. Le proprietà dell'algoritmo	2
Mod	Argomenti (l'indicazione oraria si intende stimata)	Ore								
1	Presentazione del corso, contenuti, modalità d'esame, frequenza alle lezioni, orari, modalità di esercitazione in aula, ecc. Introduzione alla Programmazione: dal problema al programma	2								
2	L'algoritmo come astrazione. Le proprietà dell'algoritmo	2								

	3	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.	5
	4	Il Problem Solving e la fase di progettazione La scomposizione del problema in sottoproblemi. Le tecniche: topdown, 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	14

		diagrammi di Nassi-Schneidermann. Il teorema di Bohm-Jacopini: enunciato. Esempi ed esercizi.	
	5	Problemi semplici e problemi complessi. Scomposizione sequenziale, selettiva, iterativa e ricorsiva. Esempi ed esercizi.	2
	6	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.	2
	7	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.	2
	8	Tipi di dati. La dichiarazione di tipo. Tassonomia dei tipi di dato. Tipi standard. Tipi semplici definiti dall'utente: enumerativo, subrange.	2
	9	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.	6
	10	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	2
	11	Esercitazione per autovalutazione: risoluzione di un problema secondo le fasi del problem solving escluso la codifica.	3
	12	Auto correzione prova di autovalutazione assistita dal docente	1
	13	La programmazione modulare. I sottoprogrammi. Utilità dei sottoprogrammi. Sottoprogramma come astrazione funzionale. Struttura risultante di un programma. La chiamata dei sottoprogrammi. Nidificazione. Esempi.	7
	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.	2
	15	I sottoprogrammi. Allocazione statica e allocazione dinamica. Il side effect. Il contour model. Esempi.	2
	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.	3

	17	Sottoprogrammi. Le procedure. Intestazione e chiamata delle procedure. Esempi ed esercizi.	2
	18	Sottoprogrammi. Le funzioni. Intestazione e chiamata delle funzioni. Procedure vs funzioni. Esempi ed esercizi.	2
	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.	3
	20	La ricorsione. Funzioni ricorsive. Gestione dell'esecuzione con le funzioni ricorsive. Esempi ed esercizi.	2
	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.	2
	22	Algoritmi di ordinamento: sort per selezione, per scambio, per inserzione. Algoritmi ricorsivi: Fattoriale.	2
	23	Struttura dei programmi C. Tipi di dati: semplici predefiniti -int, float, double e char-, enumerazione esplicita dei valori. Definizione di tipo.	2
	24	Tipi strutturati in C: il costruttore array, il costruttore struct ed il costruttore puntatore. Compatibilità dei tipi.	3
	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
	29	Esercitazione e simulazione esame scritto	3
	30	Introduzione alle applicazioni Java, all'I/O e agli operatori (capitoli 1-2)	3
	31	Introduzione ed approfondimenti su classi, oggetti, metodi e stringhe (capitoli 3-6-8)	3
	32	Istruzioni per il controllo del flusso ed operatori logici (capitoli 4-5)	3
		Esercitazione definizione classe, oggetti	3
	33	Array ed ArrayList (capitolo 7)	3
	34	Esercitazione Array e ArrayList	3
	34	Programmazione a oggetti: ereditarietà (capitolo 9)	3
	35	Esercitazione ereditarietà	3

	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.		
<b>Testi di riferimento</b>	<p>Testi da cui studiare:</p> <ul style="list-style-type: none"> <li>• Batini et al. – Fondamenti di programmazione dei Calcolatori Elettronici. Franco Angeli 1992</li> <li>• P. Deitel e H. Deitel, Il linguaggio C – Fondamenti e tecniche di programmazione - Pearson 2022 - ISBN: 9788891906243 (vanno bene anche le edizioni successive e precedenti dalla 4<sup>a</sup> in poi)</li> <li>• Programmare in Java, undicesima edizione, di Paul J. Deitel, Harvey M. Deitel, editore Pearson, ISBN: 978-8891901651 (capitoli 1-9).</li> </ul> <p>Testo integrativo, facoltativo:</p> <ul style="list-style-type: none"> <li>• Kim N. King, Programmazione in C, Maggioli Editore, ISBN 8838785821</li> </ul> <p>Gli studenti che lo desiderano possono ottenere i testi in prestito dalla Biblioteca. Può convenire verificarne la disponibilità mediante il Sistema Bibliotecario di Ateneo <a href="https://opac.uniba.it/easyweb/w8018/index.php?">https://opac.uniba.it/easyweb/w8018/index.php?</a> e contattare la biblioteca per concordare il prestito.</p>		
<b>Note ai testi di riferimento</b>	Nel corso delle lezioni la docente utilizzerà delle slide che sono disponibili sulla piattaforma ADA del dipartimento (v. sopra 'sede virtuale'). Sulla piattaforma sono anche disponibili alcune prove scritte di esami svolte.		
<b>Organizzazione della didattica</b>			
<b>Ore</b>			
Totali	Didattica frontale	Laboratorio ed esercitazioni	Studio individuale
300 ore	72 ore	45 ore	183 ore
<b>CFU/ETCS</b>			
12 CFU	9 CFU	3 CFU	

<b>Risultati di apprendimento previsti</b>	
<b>Conoscenza e capacità di comprensione</b>	<ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• Acquisire la capacità di ragionare ed individuare una soluzione ad un problema (algoritmo) secondo il paradigma della programmazione imperativa strutturata.</li> <li>• Acquisire la capacità di ragionare ed individuare una soluzione ad un problema (algoritmo) secondo il paradigma della programmazione object-oriented.</li> </ul>

<b>Metodi didattici</b>	
-------------------------	--

<b>Conoscenza e capacità di comprensione applicate</b>	<ul style="list-style-type: none"> <li>• 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;</li> <li>• Comprendere il lessico, la sintassi e la semantica del linguaggio di programmazione C;</li> <li>• Acquisire la capacità di scrivere un programma strutturato in linguaggio C;</li> <li>• Acquisire la capacità di individuare casi di test per il dominio cui fa riferimento il programma creato;</li> <li>• Acquisire la capacità di utilizzare un ambiente di sviluppo per trasformare il programma sorgente (in C) in programma eseguibile ed eseguirlo.</li> </ul>
<b>Competenze trasversali</b>	<p><b>Autonomia di giudizio</b></p> <ul style="list-style-type: none"> <li>• Acquisire la capacità di verificare che l'algoritmo individuato risponda alle specifiche di un problema;</li> <li>• Saper valutare e interpretare in maniera autonoma diverse strategie risolutive analizzando gli algoritmi proposti e fornendo soluzioni alternative.</li> <li>• Acquisire la capacità di verificare che i risultati ottenuti dopo l'esecuzione del programma siano quelli attesi.</li> </ul> <p><b>Abilità comunicative</b></p> <ul style="list-style-type: none"> <li>• Imparare a commentare il codice prodotto al fine di renderlo comprensibile e agevolmente modificabile da altri professionisti, con l'obiettivo di sviluppare in team.</li> </ul> <p><b>Capacità di apprendere in modo autonomo</b></p> <ul style="list-style-type: none"> <li>• Capacità di approfondire concetti attraverso lo studio autonomo di materiale didattico bibliografico anche attraverso piattaforme di e-learning.</li> <li>• Capacità di completare autonomamente il percorso formativo previsto dal testo di riferimento, oltre i contenuti previsti dal programma dell'insegnamento.</li> <li>• Capacità di riutilizzare le conoscenze acquisite sia in situazioni problematiche nuove, sia in contesti nuovi di programmazione imperativa.</li> </ul>
<b>Valutazione</b>	
<b>Modalità di verifica dell'apprendimento</b>	<p>Oltre alla prova di autovalutazione intermedia, che si tiene in prossimità della settimana di interruzione delle lezioni, normalmente collocata intorno alla metà di novembre, l'esame consiste in una prova di laboratorio e in una prova scritta. La prova scritta è propedeutica a quella di laboratorio.</p> <p>La prova scritta dura 60 minuti e prevede 30 domande a risposta multipla, 1 punto per ogni risposta corretta, non ci sono penalità per le risposte errate. Il voto minimo per considerare la prova superata è di 16/30.</p>
	<p>Chi supererà la prova scritta accede alla prova di laboratorio. Questa seconda prova prevede la risoluzione ed implementazione in C di un problema, come quelli di esempio riportati nella piattaforma e-learning. La durata di tale prova è di 150 minuti. Nello specifico, dato un problema vi verrà chiesto di:</p> <ul style="list-style-type: none"> <li>• Progettare la strategia di soluzione del problema individuando una scomposizione in sotto problemi e rappresentarla attraverso un albero di sviluppo.</li> <li>• Scrivere il programma in C per la soluzione della traccia, coerentemente a quanto progettato.</li> <li>• Scrivere il programma in Java per la soluzione della traccia, coerentemente a quanto progettato</li> </ul> <p>Il voto minimo per considerare la prova di laboratorio superata è di 16/30.</p> <p>Il voto finale conseguito è calcolato come media dei voti ottenuti alle due prove e viene comunicato tramite piattaforma Esse3.</p>

Criteri di valutazione	<ul style="list-style-type: none"> <li>• <b>Conoscenza e capacità di comprensione:</b> Lo studente: <ul style="list-style-type: none"> <li>○ dovrà essere in grado di analizzare problemi formulando anche ipotesi aggiuntive, individuando i dati necessari e sufficienti per la soluzione e fornendone la descrizione;</li> <li>○ 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;</li> <li>○ dovrà dimostrare di saper implementare la soluzione proposta utilizzando il linguaggio imperativo di riferimento e di saperla testare su campioni di dati;</li> <li>○ dovrà essere in grado di generalizzare soluzioni per una classe di problemi con lo stile della programmazione strutturata.</li> </ul> </li> <li><b>Conoscenza e capacità di comprensione applicate:</b> Lo studente: <ul style="list-style-type: none"> <li>○ 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;</li> <li>○ dovrà dimostrare di saper implementare la soluzione proposta utilizzando il linguaggio imperativo di riferimento e di saperla testare su campioni di dati.</li> </ul> </li> <li>• <b>Autonomia di giudizio:</b> o Lo studente: <ul style="list-style-type: none"> <li>○ dovrà essere in grado di correggere e validare il corretto funzionamento dei programmi sviluppati;</li> <li>○ dovrà essere in grado di discutere le soluzioni proposte chiarendo le scelte progettuali e implementative.</li> </ul> </li> <li>• <b>Abilità comunicative:</b> Lo studente: <ul style="list-style-type: none"> <li>○ 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;</li> <li>○ dovrà dimostrare di aver acquisito piena conoscenza dei concetti presentati a lezione nonché degli algoritmi fondamentali.</li> </ul> </li> <li>• <b>Capacità di apprendere:</b> Lo studente: <ul style="list-style-type: none"> <li>○ dovrà essere in grado di trasformare autonomamente algoritmi descritti con flowchart in programmi in linguaggio C;</li> <li>○ 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.</li> </ul> </li> </ul>											
Criteri di misurazione dell'apprendimento e di attribuzione del voto finale	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%; text-align: center;">Voto</th> <th style="text-align: center;">Descrittori</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">&lt; 18 insufficiente</td> <td>Conoscenze frammentarie e superficiali dei contenuti, errori nell'applicare i concetti, descrizione carente.</td> </tr> <tr> <td style="text-align: center;">18 - 20</td> <td>Conoscenze dei contenuti sufficienti ma generali, descrizione semplice, incertezze nell'applicazione di concetti teorici.</td> </tr> <tr> <td style="text-align: center;">21 - 23</td> <td>Conoscenze dei contenuti appropriate ma non approfondite, capacità di applicare i concetti teorici, capacità di presentare i contenuti in modo semplice.</td> </tr> <tr> <td style="text-align: center;">24 - 25</td> <td>Conoscenze dei contenuti appropriate ed ampie, discreta capacità di applicazione delle conoscenze, capacità di presentare i contenuti in modo articolato.</td> </tr> </tbody> </table>		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.
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.											

	<p>26 - 27</p> <p>28 - 29</p> <p>30 30 e lode</p>	<p>Conoscenze dei contenuti precise e complete, buona capacità di applicare le conoscenze, capacità di analisi, descrizione chiara e corretta.</p> <p>Conoscenze dei contenuti ampie, complete ed approfondite, buona applicazione dei contenuti, buona capacità di analisi e di sintesi, descrizione sicura e corretta.</p> <p>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.</p>
<p><b>Altro</b></p>	<p>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:</p> <ul style="list-style-type: none"> <li>● <a href="https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea">https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea</a></li> <li>● <a href="https://www.uniba.it/it/ricerca/dipartimenti/informatica">https://www.uniba.it/it/ricerca/dipartimenti/informatica</a></li> <li>● <a href="https://elearning.di.uniba.it/">https://elearning.di.uniba.it/</a></li> </ul> <p>I programmi degli insegnamenti sono disponibili qui:</p> <ul style="list-style-type: none"> <li>● <a href="https://programmi.di.uniba.it/">https://programmi.di.uniba.it/</a></li> </ul> <p>Le informazioni che tutti gli studenti dovrebbero conoscere sono scritte nei Regolamenti didattici e manifesti degli studi disponibili nel sito:</p> <ul style="list-style-type: none"> <li>● <a href="https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea">https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea</a></li> </ul> <p>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.</p> <p>Link al corso sulla piattaforma e-learning del dipartimento ADA:</p> <ul style="list-style-type: none"> <li>● <a href="https://elearning.di.uniba.it/">https://elearning.di.uniba.it/</a></li> </ul>	

Main information on the course	
Course name	Programmazione
Degree	Informatica e Comunicazione Digitale (Taranto)
Academic year	2023/24
European Credit Transfer and Accumulation System (ECTS), in Italian Crediti Formativi Universitari (CFU)	<p>9 CFU (each CFU corresponds to 25 hours (h) of student's time); CFU are of type T1, T2 or T3</p> <p>T1 = 8 h lecture + 17 h individual study T2 = 15 h practice + 10 h individual study T3 = 25 h individual study</p>
Settore Scientifico Disciplinare	
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	<a href="https://www.uniba.it/it/corsi/cdl-informatica-comunicazione-digitale-taranto/corso-di-laurea-triennale-in-informatica-e-comunicazione-digitale/">https://www.uniba.it/it/corsi/cdl-informatica-comunicazione-digitale-taranto/corso-di-laurea-triennale-in-informatica-e-comunicazione-digitale/</a>

Teacher(s)	
Name and Surname	Pasquale Ardimento
email	pasquale.ardimento@uniba.it
phone	
office	Department of Informatica, Via Orabona 4, 70125, Bari. Room n.569, V floor.
e-learning platform	ADA platform - <a href="https://elearning.di.uniba.it/">https://elearning.di.uniba.it/</a>
Teacher's homepage	<a href="https://www.uniba.it/it/docenti/ardimento-pasquale">https://www.uniba.it/it/docenti/ardimento-pasquale</a>
Office hours	Wednesday 2.30pm – 3.30pm, or on another day by appointment by email

Syllabus	
Course goals	The course aims to introduce the methodology of problem-solving 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, identify algorithmic solution strategies, and use the C programming language to implement them.		
<b>Prerequisites/requirements</b>	No prerequisites are required, as this is a course taught in the first semester of the first year.		
<b>Course program</b>	<b>Mod</b>	<b>Topics (the indicate time is an estimate)</b>	<b>Hours</b>
	1	Presentazione del corso, contenuti, modalità d'esame, frequenza alle lezioni, orari, modalità di esercitazione in aula, ecc. Introduzione alla Programmazione: dal problema al programma	2
	2	L'algoritmo come astrazione. Le proprietà dell'algoritmo	2
	3	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.	5
	4	Il Problem Solving e la fase di progettazione La scomposizione del problema in sottoproblemi. Le tecniche: topdown, bottom-up, ibrida. La costruzione del metodoolutivo. La rappresentazione dell'algoritmo mediante linguaggi di rappresentazione: i diagrammi di flusso strutturati, il linguaggio lineare, l'albero di decomposizione, i	14
		diagrammi di Nassi-Schneidermann. Il teorema di Bohm-Jacopini: enunciato. Esempi ed esercizi.	
	5	Problemi semplici e problemi complessi. Scomposizione sequenziale, selettiva, iterativa e ricorsiva. Esempi ed esercizi.	2
	6	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.	2
	7	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.	2
	8	Tipi di dati. La dichiarazione di tipo. Tassonomia dei tipi di dato. Tipi standard. Tipi semplici definiti dall'utente: enumerativo, subrange.	2
	9	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.	6
	10	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	2
	11	Esercitazione per autovalutazione: risoluzione di un problema secondo le fasi del problem-solving escluso la codifica.	3
12	Auto correzione prova di autovalutazione assistita dal docente	1	

13	La programmazione modulare. I sottoprogrammi. Utilità dei sottoprogrammi. Sottoprogramma come astrazione funzionale. Struttura risultante di un programma. La chiamata dei sottoprogrammi. Nidificazione. Esempi.	7	
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.	2	
15	I sottoprogrammi. Allocazione statica e allocazione dinamica. Il side effect. Il contour model. Esempi.	2	
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.	3	
17	Sottoprogrammi. Le procedure. Intestazione e chiamata delle procedure. Esempi ed esercizi.	2	
18	Sottoprogrammi. Le funzioni. Intestazione e chiamata delle funzioni. Procedure vs funzioni. Esempi ed esercizi.	2	
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.	3	
20	La ricorsione. Funzioni ricorsive. Gestione dell'esecuzione con le funzioni ricorsive. Esempi ed esercizi.	2	
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.	2	
22	Algoritmi di ordinamento: sort per selezione, per scambio, per inserzione. Algoritmi ricorsivi: Fattoriale.	2	
23	Struttura dei programmi C. Tipi di dati: semplici predefiniti -int, float, double e char-, enumerazione esplicita dei valori. Definizione di tipo.	2	
24	Tipi strutturati in C: il costruttore array, il costruttore struct ed il costruttore puntatore. Compatibilità dei tipi.	3	
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	
29	Esercitazione e simulazione esame scritto	3	
30	Introduzione alle applicazioni Java, all'I/O e agli operatori (capitoli 1-2)	3	
31	Introduzione ed approfondimenti su classi, oggetti, metodi e stringhe (capitoli 3-6-8)	3	
32	Istruzioni per il controllo del flusso ed operatori logici (capitoli 4-5)	3	
33	Esercitazione definizione classe, oggetti	3	

	34	Array ed ArrayList (capitolo 7)	3	
	35	Esercitazione Array e ArrayList	3	
<b>Books of reference</b>	<p>Books for studying:</p> <ul style="list-style-type: none"> <li>Batini et al. – *Fondamenti di programmazione dei Calcolatori Elettronici*. Franco Angeli, 1992</li> <li>P. Deitel e H. Deitel, Il linguaggio C – Fondamenti e tecniche di programmazione - 9th edition Pearson 2022 - ISBN: 9788891906243 (later and earlier editions from the 4th edition onward are also acceptable)</li> <li>Programmare in Java, 11th edition, di Paul J. Deitel, Harvey M. Deitel, Pearson, ISBN: 978-8891901651 (capitoli 1-9).</li> <li></li> </ul> <p>Additional text:</p> <ul style="list-style-type: none"> <li>Kim N. King, *Programmazione in C*, Maggioli Editore, ISBN 8838785821</li> </ul> <p>Students who wish can borrow the books from the Library. It may be useful to check availability through the University Library System at <a href="https://opac.uniba.it/easyweb/w8018/index.php">https://opac.uniba.it/easyweb/w8018/index.php</a> and contact the library to arrange the loan.</p>			
<b>Notes to the books</b>	<p>During the lessons the teacher will use slides that retrace the contents of the book, therefore, they will not be provided. The reference text contains all the topics of the course, therefore, it is advisable to study from the text and to carry out independently and constantly all the exercises included at the end of each chapter covered in class.</p> <p>On the department's e-learning platform (see above 'virtual office') are available:</p> <ul style="list-style-type: none"> <li>supporting video material used in class;</li> <li>some traces of written tests of exams, with examples of traces carried out;</li> </ul>			
<b>Organization of the didactic activities</b>				
<b>Hours</b>				
Total	Lectures	Practice sessions	Project work	Individual study
300 hours	72 hours	45 hours	00 hours	183 hours
<b>CFU/ETCS</b>				
12 CFU	09 CFU	03 CFU	00 CFU	

<b>Teaching methods</b>	
	Lectures, exercises and autonomous and group activities in the classroom and at home (as detailed in the program). Non-attending students can work individually by making arrangements with the teacher.

<b>Expected learning outcomes</b>	
-----------------------------------	--

<b>Knowledge and understanding</b>	<ul style="list-style-type: none"> <li>• Acquire knowledge of the theoretical and practical aspects related to designing problem-solving solutions using a computer, as well as an understanding of the tools used for programming.</li> <li>• Gain an understanding of how to instruct a computer (a universal automatic machine, hardware) to solve a problem or a class of problems that it can handle, using an appropriate method and language, by creating a specific program (software) executable by the computer.</li> <li>• Develop the ability to reason and identify a solution to a problem (algorithm) according to the structured imperative programming paradigm.</li> <li>• Develop the ability to reason and identify a solution to a problem (algorithm) according to the Object-Oriented programming paradigm</li> </ul>
<b>Applying knowledge and understanding</b>	<ul style="list-style-type: none"> <li>• Understand the use of a non-conventional design language (e.g., pseudocode) and the use of a graphical representation (e.g., flowchart) to describe an algorithm with simple formalism.</li> <li>• Understand the lexicon, syntax, and semantics of the C programming language.</li> <li>• Develop the ability to write a structured program in C.</li> <li>• Acquire the ability to identify test cases for the domain to which the created program applies.</li> <li>• Gain the ability to use a development environment to transform the source program (in C) into an executable program and run it.</li> </ul>
<b>Other skills</b>	<p><i>Making judgements</i></p> <ul style="list-style-type: none"> <li>• Develop the ability to verify that the identified algorithm meets the problem specifications.</li> <li>• Be able to evaluate and interpret different solution strategies independently by analyzing proposed algorithms and providing alternative solutions.</li> <li>• Acquire the ability to check that the results obtained after executing the program are as expected.</li> </ul> <p><i>Communication</i></p> <ul style="list-style-type: none"> <li>• Learn to comment on the produced code to make it understandable and easily modifiable by other professionals, with the aim of developing in a team.</li> </ul> <p><i>Learning skills</i></p> <ul style="list-style-type: none"> <li>• Ability to deepen concepts through independent study of bibliographic educational material, including e-learning platforms.</li> <li>• Ability to independently complete the educational path outlined by the reference text, beyond the content covered by the course syllabus.</li> <li>• Ability to reuse acquired knowledge in both new problem situations and new contexts of imperative programming.</li> </ul>

<b>Assessment</b>	
<b>Assessment methods</b>	<p>In addition to the mid-term self-assessment, which takes place around the week of the mid-semester break, typically around mid-November, the exam consists of a laboratory test and a written test. The written test is a prerequisite for the laboratory test.</p>

	<p>The written test lasts 60 minutes and consists of 30 multiple-choice questions, with 1 point for each correct answer; there are no penalties for incorrect answers. The minimum passing grade for the written test is 16/30.</p> <p>Those who pass the written test will proceed to the laboratory test. This second test involves solving and implementing a problem in C, like the examples provided on the e-learning platform. The duration of this test is 150 minutes. Specifically, given a problem, you will be asked to:</p> <ul style="list-style-type: none"> <li>• Design a solution strategy by breaking down the problem into sub-problems and represent it using a development tree.</li> <li>• Write the C program to solve the problem as designed.</li> <li>• Write the Java program to solve the problem as designed.</li> </ul> <p>The minimum passing grade for the laboratory test is 16/30.</p> <p>The final grade is calculated as the average of the grades obtained in the two tests and is communicated via the Esse3 platform.</p>
<p><b>Evaluation criteria</b></p>	<ul style="list-style-type: none"> <li>• Knowledge and understanding: <ul style="list-style-type: none"> <li>- The student must be able to analyze problems, even formulating additional hypotheses, identifying the necessary and sufficient data for the solution, and providing a description.</li> <li>- They must be able to identify a solution strategy that involves breaking down the problem into sub-problems and representing both the breakdown and the algorithms with appropriate descriptive languages.</li> <li>- They must demonstrate the ability to implement the proposed solution using the reference imperative language and test it on data samples.</li> <li>- They must be able to generalize solutions for a class of problems using structured programming.</li> </ul> </li> <li>• Applied knowledge and understanding: <ul style="list-style-type: none"> <li>- They must be able to identify a solution strategy that involves breaking down the problem into sub-problems and representing both the breakdown and the algorithms with appropriate descriptive languages.</li> <li>- They must demonstrate the ability to implement the proposed solution using the reference imperative language and test it on data samples.</li> </ul> </li> <li>• Autonomy of judgment: <ul style="list-style-type: none"> <li>- The student must be able to correct and validate the correct functioning of the developed programs.</li> <li>- They must be able to discuss the proposed solutions, explaining the design and implementation choices.</li> </ul> </li> <li>• Communication skills: <ul style="list-style-type: none"> <li>- The student must be able to make the written code understandable to others by providing a general description and specific comments on the instructions and control structures used.</li> <li>- They must demonstrate full knowledge of the concepts presented in the lessons as well as the fundamental algorithms.</li> </ul> </li> <li>• Learning skills: <ul style="list-style-type: none"> <li>- The student must be able to independently transform algorithms described with flowcharts into C programs.</li> <li>- The student must be able to use alternative solutions described in the reference text, even if not covered during lessons, such as the different methods of declaring variables.</li> </ul> </li> </ul>
<p>Measurements and final grade</p>	<ul style="list-style-type: none"> <li>• &lt; 18 <ul style="list-style-type: none"> <li>○ Insufficient: Fragmented and superficial knowledge of the content, errors in applying concepts, inadequate description.</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>• 18 - 20 <ul style="list-style-type: none"> <li>○ Sufficient: General knowledge of the content, simple description, uncertainty in applying theoretical concepts.</li> </ul> </li> <li>• 21 - 23 <ul style="list-style-type: none"> <li>○ Adequate: Appropriate but not in-depth knowledge of the content, ability to apply theoretical concepts, ability to present content in a simple manner.</li> </ul> </li> <li>• 24 - 25 <ul style="list-style-type: none"> <li>○ Good: Appropriate and broad knowledge of the content, reasonable ability to apply knowledge, ability to present content in a structured manner.</li> </ul> </li> <li>• 26 - 27 <ul style="list-style-type: none"> <li>○ Very Good: Precise and complete knowledge of the content, good ability to apply knowledge, analytical skills, clear and correct description.</li> </ul> </li> <li>• 28 – 29 <ul style="list-style-type: none"> <li>○ Excellent: Broad, complete, and in-depth knowledge of the content, good application of concepts, strong analytical and synthesis skills, confident and correct description.</li> </ul> </li> <li>• 30 - 30 with honors <ul style="list-style-type: none"> <li>○ Outstanding: Very broad, complete, and in-depth knowledge of the content, well-established ability to apply concepts, excellent analytical, synthesis, and interdisciplinary connection skills, mastery in description.</li> </ul> </li> </ul>
<p><b>Further information</b></p>	<p>It is recommended that students rely exclusively on the information and communications provided on the official websites of the Department of Computer Science or on social groups only if they are created and managed solely by the instructors of the respective courses:</p> <ul style="list-style-type: none"> <li>• <a href="https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-dilaurea/corsi-di-laurea">https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-dilaurea/corsi-di-laurea</a></li> <li>• <a href="https://www.uniba.it/it/ricerca/dipartimenti/informatica">https://www.uniba.it/it/ricerca/dipartimenti/informatica</a></li> <li>• <a href="https://elearning.di.uniba.it/">https://elearning.di.uniba.it/</a></li> </ul> <p>Course syllabi are available here:</p> <ul style="list-style-type: none"> <li>• <a href="https://programmi.di.uniba.it/">https://programmi.di.uniba.it/</a></li> </ul> <p>Information that all students should be aware of is written in the Academic Regulations and study guides available on the website:</p> <ul style="list-style-type: none"> <li>• <a href="https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-dilaurea/corsi-di-laurea">https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-dilaurea/corsi-di-laurea</a></li> </ul> <p>Students are advised to be cautious of information and materials circulating on unofficial websites or social groups, as they are often unreliable, incorrect, or incomplete. For any doubts, please request a meeting with the instructor according to the office hours procedures.</p> <p>Link to the course on the department's ADA e-learning platform:</p> <ul style="list-style-type: none"> <li>• <a href="https://elearning.di.uniba.it/">https://elearning.di.uniba.it/</a></li> </ul>