Principali informazioni sull'insegnamento

Denominazione dell'insegnamento	**Programmazione (track cognomi M-Z)**
Corso di studio	Informatica
Anno Accademico	2022/23
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	https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/informatica-270/laurea-triennale-in-informatica-d.m.-270-1

Docente/i

Nome e cognome	Teresa ROSELLI
Indirizzo mail	Teresa.roselli@uniba.it
Telefono	080-5443276
Sede	Dipartimento di Informatica, Via Orabona 4, 70125, Bari. Stanza n.772, 7^ piano.
Sede virtuale	Piattaforma ADA - https://elearning.di.uniba.it/
Sito web del docente	https://www.uniba.it/it/docenti/roselli-teresa
Ricevimento	Lunedì e Martedì 13:30 – 15:00, o in altro giorno previo appuntamento per email

Syllabus

Obiettivi formativi	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.
Prerequisiti	Non è richiesto alcun prerequisito in quanto trattasi di un insegnamento del primo semestre del primo anno

Contenuti di insegnamento (Programma)

Mod	**Argomenti** (l'indicazione oraria si intende stimata)	**Ore**
1	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	3
2	L'algoritmo come astrazione. Le proprietà dell'algoritmo	3
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.	6
4	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	15

			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.	3
		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.	3
		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.	3
		8	Tipi di dati. La dichiarazione di tipo. Tassonomia dei tipi di dato. Tipi standard. Tipi semplici definiti dall'utente: enumerativo, subrange.	3
		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.	7
		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	3
		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
29	Esercitazione e simulazione esame scritto	3

Testi di riferimento	Testi da cui studiare: Batini et al. – Fondamenti di programmazione dei Calcolatori Elettronici. Franco Angeli 1992 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: Kim N. King, Programmazione in C, Maggioli Editore, ISBN 8838785821 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 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.

Organizzazione della didattica

Ore

Totali	Didattica frontale	Laboratorio ed esercitazioni	Studio individuale
300 ore	72 ore	45 ore	183 ore

CFU/ETCS

12 CFU	9 CFU	3 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 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.
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	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 orale. La prova di laboratorio è propedeutica a quella orale. La prova scritta, che avviene nei laboratori di informatica, consiste nella soluzione di un problema applicando le fasi del problem solving compresa la codifica nel linguaggio C. La prova orale consiste nella visione della prova scritta e degli eventuali errori commessi e nell'esposizione, in risposta a domande, riguardante la parte teorica trattata durante il corso.

<table>
<tr><td></td><td>Il superamento della prova scritta e la conseguente ammissione alla prova orale vengono comunicati con lista pubblica sulla piattaforma ADA. Il voto finale conseguito viene proposto esclusivamente sulla piattaforma Esse3.</td></tr>
<tr><td>Criteri di valutazione</td><td>

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.
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>
<table>
<tr><th>Voto</th><th>Descrittori</th></tr>
<tr><td>< 18
insufficiente</td><td>Conoscenze frammentarie e superficiali dei contenuti, errori nell’applicare i concetti, descrizione carente.</td></tr>
<tr><td>18 - 20</td><td>Conoscenze dei contenuti sufficienti ma generali, descrizione semplice, incertezze nell’applicazione di concetti teorici.</td></tr>
<tr><td>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>
</table>
</td></tr>
</table>

	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.

Altro

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:

- 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/

I programmi degli insegnamenti sono disponibili qui:

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

Le informazioni che tutti gli studenti dovrebbero conoscere sono scritte nei Regolamenti didattici e manifesti degli studi disponibili nel sito:

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