Principali informazioni sull'insegnamento	
Denominazione dell'insegnamento	**Laboratorio di Informatica**
Corso di studio	Informatica
Anno di corso	Primo
Crediti formativi universitari (CFU) / European Credit Transfer and Accumulation System (ECTS):	9+3
SSD	INF/01
Lingua di erogazione	Italiano
Periodo di erogazione	Secondo semestre
Obbligo di frequenza	No

Docente	
Nome e cognome	Fabio Abbattista
Indirizzo mail	Fabio.abbattista@uniba.it
Telefono	
Sede	Dipartimento di informatica – Universita' di Bari
Sede virtuale	
Ricevimento (giorni, orari e modalità)	Martedi 15.00-16.00

Syllabus	
Obiettivi formativi	• *Conoscenza e capacità di comprensione* Lo studente dovrà essere in grado di analizzare e risolvere problemi di media complessita', progettando e sviluppando programmi in un linguaggio di programmazione di alto livello. • *Conoscenza e capacità di comprensione applicate* Lo studente dovrà acquisire competenze relative a: - Traduzione di semplici algoritmi in programmi correttamente funzionanti e ben documentati; - Utilizzo di tecniche di programmazione difensiva, per limitare l'introduzione di malfunzionamenti nei programmi; - Verifica empirica della correttezza dei programmi mediante testing; - Capacità di problem-solving attraverso l'applicazione di nozioni apprese nelle discipline informatiche di base nella pratica della programmazione. • *Autonomia di giudizio* Lo studente deve dimostrare di aver acquisito autonomia di giudizio e capacità di valutazione degli algoritmi sviluppati da lui o da terzi. • *Capacità di apprendere* Lo studente dovrà mostrare di essere in grado di orientarsi agevolmente nelle problematiche relative alla comprensione e all'utilizzo delle tecnologie e dei

	metodi di competenza per lo sviluppo di algoritmi e per la loro traduzione in programmi per computer.
Prerequisiti	Buona conoscenza della lingua inglese
Contenuti di insegnamento (Programma)	1. Stili di programmazione Motivazioni Uso appropriato dei nomi Scrittura appropriata di espressioni e istruzioni Consistenza ed espressioni idiomatiche Commenti Convenzioni di programmazione 2. Testing e Debugging Bug Tecniche di debugging Strumenti per il debugging Generalità sul testing Il test di unità Tecniche di testing cUnit 3. Programmazione modulare Modularizzazione e strutturazione dei programmi Strutturazione dei file sorgente Strutturazione di progetti in Eclipse CDT 4. Documentazione del codice Generalità sulla documentazione di codice in linea Documentazione automatica di codice Doxygen 5. Puntatori Procedure e funzioni I/O Memoria dinamica
Testi di riferimento	P. Deitel e H. Deitel, Il Linguaggio C - Fondamenti e tecniche di programmazione, Pearson, 2013 A. Downey, Pensare in Python – Come pensare da informatico, O'Reilly, 2018
Note ai testi di riferimento	Testi integrativi: B.W. Kernighan e R. Pike, Programmazione nella pratica, Addison-Wesley, 1999. J.R. Hanly, E.B. Koffman, Problem solving e programmazione in C, Apogeo, 2013

Organizzazione della didattica			
Ore			
Totali	Didattica frontale	Pratica (laboratorio, campo, esercitazione, altro)	Studio individuale
69	*24*	*45*	*81*
CFU/ETCS			

6	*3*	*3*	

Metodi didattici	Learning by doing

Risultati di apprendimento previsti	
Conoscenza e capacità di comprensione	Lo studente dovrà essere in grado di analizzare e risolvere problemi di media complessita', progettando e sviluppando programmi in un linguaggio di programmazione di alto livello.
Conoscenza e capacità di comprensione applicate	Lo studente dovrà acquisire competenze relative a: - Traduzione di semplici algoritmi in programmi correttamente funzionanti e ben documentati; - Utilizzo di tecniche di programmazione difensiva, per limitare l'introduzione di malfunzionamenti nei programmi; - Verifica empirica della correttezza dei programmi mediante testing; - Capacità di problem-solving attraverso l'applicazione di nozioni apprese nelle discipline informatiche di base nella pratica della programmazione.
Competenze trasversali	• *Autonomia di giudizio* Lo studente deve dimostrare di aver acquisito autonomia di giudizio e capacità di valutazione degli algoritmi sviluppati da lui o da terzi. • *Abilità comunicative* Lo studente deve essere in grado di illustrare in modo appropriato le caratteristiche tecniche degli strumenti e delle metodologie informatiche apprese nel corso del primo anno di corso, relative allo sviluppo di programmi di media complessita'. • *Capacità di apprendere* Lo studente dovrà mostrare di essere in grado di orientarsi agevolmente nelle problematiche relative alla comprensione e all'utilizzo delle tecnologie e dei metodi di competenza per lo sviluppo di algoritmi e per la loro traduzione in programmi per computer.

Valutazione	
Modalità di verifica dell'apprendimento	Realizzazione in itinere di un progetto, in gruppo. Il superamento della prova in itinere attribuisce una premialità sul voto finale. L'esame consiste di una prova pratica e in una prova orale individuale.
Criteri di valutazione	Lo studente dovrà dimostrare di aver acquisito la capacità di progettare l'algoritmo ottimale per la soluzione di problemi con caratteristiche diverse. Inoltre deve aver sviluppato buone competenze nell'utilizzo di un linguaggio di programmazione di alto livello. L'esame finale consiste nella risoluzione di un problema, identificando l'algoritmo e sviluppando il corrispondente programma (per chi non e' stato esonerato) e in una prova orale.
Criteri di misurazione	

dell'apprendimento e di attribuzione del voto finale	
Altro	

General information	
Academic subject	
Degree course	Computer science
Academic Year	First year
European Credit Transfer and Accumulation System (ECTS)	9+3
Language	Italian
Academic calendar (starting and ending date)	March 1, 2023 – January 7, 2023
Attendance	Not mandatory

Professor/ Lecturer	
Name and Surname	Fabio Abbattista
E-mail	Fabio.abbattista@uniba.it
Telephone	
Department and address	Dipartimento di Informatica, Universita' di bari
Virtual headquarters	
Tutoring (time and day)	Tuesday, 15.00-16.00

Syllabus	
Learning Objectives	• Knowledge and understanding The student should be able to analyze and solve medium complexity problems, designing and developing programs in an high-level programming language. • Applied knowledge and understanding The student will have to acquire skills related to: - Translation of simple algorithms into properly functioning programs; - Ability to use defensive programming technique, to reduce the number of program failures. - Empirical verification of programs correctness by using software testing. - Problem-solving skills through the application of notions learned in basic computer disciplines in the practice of programming. • Autonomy of judgment The student must demonstrate that he has acquired autonomy of judgment and evaluation skills in the context of the development of algorithms. • Communicating knowledge and understanding The student must be able to present appropriately technical features of tools and methods learned in the first year courses, concerning the development of medium complexity computer programs. • Ability to learn The student must show that they have developed the ability to learn and to orient themselves easily in the problems related to the understanding and use of information technologies in its specific field of application.
Course prerequisites	Good understanding of the English language.
Contents	1. Programming styles

	Why using styles Proper use of names Proper writing of expressions and instructions Consistency and idiomatic expressions Comments Programming conventions 2. Testing and Debugging Bug Debugging techniques Debugging tools General information on testing Unit testing Testing techniques cUnit 3. Modular programming Modularization and structuring of programs Structuring of source files Structuring of projects in Eclipse CDT 4. Code documentation General information on online code documentation Automatic code documentation Doxygen 5. Pointers Procedures and functions I/O Dynamic memory
Books and bibliography	P. Deitel e H. Deitel, Il Linguaggio C - Fondamenti e tecniche di programmazione, Pearson, 2013. A. Downey, Pensare in Python – Come pensare da informatico, O'Reilly, 2018
Additional materials	B.W. Kernighan e R. Pike, Programmazione nella pratica, Addison-Wesley, 1999. J.R. Hanly, E.B. Koffman, Problem solving e programmazione in C, Apogeo, 2013

Work schedule			
Total	Lectures	Hands on (Laboratory, working groups, seminars, field trips)	Out-of-class study hours/ Self-study hours
Hours			
69	*24*	*45*	*81*
ECTS			
6	*3*	*3*	
Teaching strategy	*Learning by doing*		
Expected learning outcomes			

Knowledge and understanding on:	The student should be able to analyze and solve medium complexity problems, designing and developing programs in a high-level programming language.
Applying knowledge and understanding on:	The student will have to acquire skills related to: - Translation of simple algorithms into properly functioning programs; - Ability to use defensive programming technique, to reduce the number of program failures. - Empirical verification of programs correctness by using software testing. - Problem-solving skills through the application of notions learned in basic computer disciplines in the practice of programming.
Soft skills	• Autonomy of judgment The student must demonstrate that he has acquired autonomy of judgment and evaluation skills in the context of the development of algorithms. • Communicating knowledge and understanding The student must be able to present appropriately technical features of tools and methods learned in the first year courses, concerning the development of medium complexity computer programs. • Ability to learn The student must show that they have developed the ability to learn and to orient themselves easily in the problems related to the understanding and use of information technologies in its specific field of application.

Assessment and feedback	
Methods of assessment	Development of e software project, during the semester. The grade obtained award a bonus on the final grade. The final exam consists of an individual practical test.
Evaluation criteria	The student will have to demonstrate that he has acquired the ability to design an algorithm for solving problems with different characteristics. In addition, he must have developed good skills in the use of an high-level programming language. • The final exam consists in solving a problem by identifying the algorithm and developing the related program (for those who have not taken / passed the exemption tests) and in the oral.
Criteria for assessment and attribution of the final mark	
Additional information	