Principali informazioni sull'insegnamento	
Titolo insegnamento	Linguaggi di programmazione
Corso di studio	Informatica e Tecnologie per la Produzione del Software
Crediti formativi	7+2
Denominazione inglese	Programming languages
Obbligo di frequenza	No
Lingua di erogazione	Italiano

Docente responsabile	Nome Cognome	Indirizzo Mail
	Emanuele Covino	emanuele.covino@uniba.it
Luogo e orario di ricevimento	Dipartimento Informatica	Martedì dalle 11:00 – 12:00

Dettaglio credi formativi	Ambito disciplinare	SSD	Crediti
	Informatico, fisico, matematico, economico, linguistico	INF/01-Informatica	7+2

Modalità di erogazione	
Periodo di erogazione	II semestre
Anno di corso	I
Modalità di erogazione	Lezioni frontali, esercitazioni

Organizzazione della didattica	
Ore totali	86 (corso) + 139 (studio individuale)
Ore di corso	56+30
Ore di studio individuale	119+20

Calendario (indicativo)	
Inizio attività didattiche	25 Febbraio
Fine attività didattiche	31 Maggio

Syllabus	
Prerequisiti	
Risultati di apprendimento previsti	• *Conoscenza e capacità di comprensione* Lo studente dovrà acquisire competenze relative alle caratteristiche dei linguaggi di programmazione dal punto di vista sintattico, semantico, implementativo e pragmatico, anche nel contesto di diversi paradigmi di programmazione. • *Conoscenza e capacità di comprensione applicate* Lo studente dovrà acquisire competenze relative alla progettazione, implementazione e funzionamento dei compilatori. • *Autonomia di giudizio* Lo studente deve dimostrare di aver acquisito autonomia di giudizio e di capacità di valutazione nell'ambito dell'utilizzo dei linguaggi di programmazione. • *Abilità comunicative* Lo studente deve essere in grado di illustrare in modo

	appropriato le caratteristiche tecniche degli strumenti e delle metodologie informatiche relative ai linguaggi di programmazione. • *Capacità di apprendere* Lo studente dovrà mostrare di aver sviluppato capacità di apprendere e di orientarsi agilmente nelle problematiche relative alla comprensione e all'utilizzo delle tecnologie informatiche nel suo specifico campo di applicazione.
Contenuti di insegnamento	MACCHINE ASTRATTE: La nozione di macchina astratta e l'interprete. Interprete. Un esempio di macchina astratta: la macchina hardware. Implementazione di un linguaggio. Realizzazione di una macchina astratta. Implementazione: il caso ideale. Implementazione: il caso reale e la macchina intermedia. Gerarchie di macchine astratte. DESCRIVERE UN LINGUAGGIO DI PROGRAMMAZIONE: Livelli di descrizione. Grammatica e sintassi. Vincoli sintattici contestuali. Compilatori. Semantica. Pragmatica. Implementazione. LINGUAGGI REGOLARI PER L' ANALISI LESSICALE: Token. Linguaggi formali e operazioni. Espressioni regolari. Automi finiti: DFA e NFA. equivalenza. Da espressioni regolari ad automi finiti. Automi finiti e grammatiche. Minimizzare un DFA. Generatori di analizzatori lessicali. Dimostrare che un linguaggio non è regolare. LINGUAGGI LIBERI PER L'ANALISI SINTATTICA: Linguaggi, derivazioni e alberi. Automi a pila. Dimostrare che un linguaggio non è libero. Oltre i linguaggi liberi. Analizzatori sintattici Manipolazioni delle grammatiche. Parser top-down. Parser a discesa ricorsiva. First e Follow. Grammatiche LL(1). Parser LL(1) non ricorsivi. Grammatiche LL(k). Parser bottom-up e Parsing con grammatiche ambigue. Linguaggi e grammatiche deterministici. Generatori di analizzatori sintattici. FONDAMENTI: Il problema della fermata. Espressività dei linguaggi di programmazione. Formalismi per la calcolabilità. Funzioni e algoritmi. I NOMI E L'AMBIENTE: Nomi e oggetti denotabili. Oggetti denotabili. Ambiente e blocchi. I blocchi. Tipi di ambiente. Operazioni sull'ambiente. Regole di scope. Scope statico. Scope dinamico. Problemi di scope. LA GESTIONE DELLA MEMORIA: Tecniche di gestione della memoria. Gestione statica della memoria. Gestione dinamica mediante pila. Record di attivazione per i blocchi in-line. Record

	di attivazione per le procedure. Gestione della pila. Gestione dinamica mediante heap. Blocchi di dimensione fissa. Blocchi di dimensione variabile. Implementazione delle regole di scope. Scope statico: la catena statica. il display. Scope dinamico: lista di associazioni e CRT. STRUTTURARE IL CONTROLLO: Le espressioni. Sintassi e Semantica delle espressioni. Valutazione delle espressioni. La nozione di comando. La variabile. L'assegnamento. Comandi per il controllo di sequenza. Comandi per il controllo di sequenza esplicito. Comandi condizionali. Comandi iterativi. Programmazione strutturata. La ricorsione. La ricorsione in coda. Ricorsione e iterazione. ASTRARRE SUL CONTROLLO: Sottoprogrammi. Astrazione funzionale. Passaggio dei parametri. Funzioni di ordine superiore. Funzioni come parametro. Funzioni come risultato. Eccezioni. Implementare le eccezioni. STRUTTURARE I DATI: Tipi di dato. Tipi come supporto all'organizzazione concettuale. Tipi per la correttezza. Tipi e implementazione. Sistemi di tipi. Controlli statici e dinamici. Tipi scalari. Booleani. Caratteri. Interi. Reali. Virgola fissa. Complessi. Void. Enumerazioni. Intervalli. Tipi ordinali. Tipi composti. Record. Record varianti e unioni. Array. Insiemi. Puntatori. Tipi ricorsivi. Funzioni. Equivalenza: per nome, strutturale. Compatibilità e conversione. Polimorfismo. Overloading. Polimorfismo universale parametrico. Polimorfismo universale di sottotipo. Cenni sull'implementazione. Controllo e inferenza di tipo. Sicurezza: Evitare i dangling reference. Tombstone, Lucchetti e chiavi. Garbage collection: Contatori dei riferimenti, Mark and sweep, rovesciare i puntatori, Mark and compact, Copia. Il Garbage collector di Java. ASTRARRE SUI DATI: Tipi di dato astratti. Nascondere l'informazione. Indipendenza dalla rappresentazione. Moduli IL PARADIGMA ORIENTATO AGLI OGGETTI: Concetti fondamentali: Limiti degli ADT. Oggetti. Classi. Incapsulamento. Meccanismi di delega. Memorizzazione. Sottotipi e gerarchia. Ereditarietà. Selezione dinamica dei metodi. Aspetti implementativi. Polimorfismo di sottotipo ed universale. Tipi generici in Java. Array. Covarianza e controvarianza. Esercizi pratici in Java/C++ su classi, astrazioni dati, ereditarietà. IL PARADIGMA FUNZIONALE: Introduzione. Computazione senza stato. Valutazione. Ambienti. Tipi. Oggetti infiniti. Aspetti imperativi.

	LINGUAGGIO C++ (Deitel & Deitel o qualsiasi altro manuale) Classi e astrazioni dati: Definizione delle strutture; Accedere ai membri; Implementare un astract data type TIME con una classe; Scope della classe e accesso ai membri; Separare l'interfaccia dall'implementazione; Controllare l'accesso ai membri; Funzioni di accesso e di utility; Inizializzare gli oggetti: costruttori; Distruttori ; Quando sono chiamati costruttori e distruttori; Usare dati membro e funzioni membro; Ritornare una reference a una dato membro privato; Assegnamento di default per copia. Classi: Oggetti const e funzioni membro const; Composizione: oggetti come membri di classi; Funzioni friend e classi friend ; Uso del puntatore this; Allocazione dinamica della memoria con new e delete; Membri static ; Data abstraction e information hiding; Esempio: dato astratto queue. Ereditarietà:classi base e classi derivate; Membri protetti; Cast di puntatori a classi base verso puntatori a classi derivate; Uso delle funzioni membro; Override dei membri della classe base in una classe derivata; Public, Protected e Private ; Classi base dirette e indirette; Usare costruttori e distruttori nelle classi derivate; Conversione di oggetti impliciti di classi derivate a oggetti di classi base; Composizione vs. Ereditarietà; Relazione "Uses A" e "Knows A"; Point, Circle, Cylinder. Eccezioni.

Programma	
Testi di riferimento	Gabbrielli, Martini - Linguaggi di Programmazione, McGraw-Hill. 2^ edizione
Note ai testi di riferimento	Testi integrativi: Ausiello, D'amore, Gambosi - Linguaggi, modelli, complessità – II ed., Utet Aho, Lam, Sethi, Ullman – Compilatori: principi, tecniche e strumenti, II ed., Pearson
Metodi didattici	Lezioni frontali ed esercitazioni pratiche.
Metodi di valutazione	Un prova scritta in itinere, non obbligatoria, da tenersi durante l'interruzione delle lezioni prevista, che verte sugli argomenti trattati nella prima parte del corso. Il superamento della prova in itinere e/o i risultati delle esercitazioni pratiche attribuiscono una premialità sul voto finale. Prova scritta e orale.
Criteri di valutazione	Lo studente dovrà dimostrare di aver acquisito la capacità di individuare il linguaggio di programmazione appropriato per la soluzione di problemi con caratteristiche diverse, valutandone gli aspetti sintattici, semantici, implementativi, pragmatici.

Altro	