UNIVERSITÀ DEGLI STUDI DI PALERMO

DIPARTIMENTO	Matematica e Informatica
ANNO ACCADEMICO OFFERTA	2021/2022
ANNO ACCADEMICO EROGAZIONE	2023/2024
CORSO DILAUREA	INFORMATICA
INSEGNAMENTO	COMPILATORI
TIPO DI ATTIVITA'	B
AMBITO	50166-Discipline Informatiche
CODICE INSEGNAMENTO	14049
SETTORI SCIENTIFICO-DISCIPLINARI	INF/01
DOCENTE RESPONSABILE	MANTACI SABRINA Professore Associato Univ. di PALERMO
ALTRI DOCENTI	
CFU	6
NUMERO DI ORE RISERVATE ALLO STUDIO PERSONALE	102
NUMERO DI ORE RISERVATE ALLA DIDATTICA ASSISTITA	48
PROPEDEUTICITA'	05880 - PROGRAMMAZIONE E LABORATORIO C.I. 16670 - ALGORITMI E STRUTTURE DATI 16784 - SISTEMI OPERATIVI 16450 - ARCHITETTURE DEGLI ELABORATORI 16671 - INFORMATICA TEORICA
MUTUAZIONI	
ANNO DI CORSO	3
PERIODO DELLE LEZIONI	2° semestre
MODALITA' DI FREQUENZA	Facoltativa
TIPO DI VALUTAZIONE	Voto in trentesimi
ORARIO DI RICEVIMENTO DEGLI STUDENTI	**MANTACI SABRINA** Giovedì 15:00 18:00 Room 217 - second Floor. DMI - Via Archirafi 34

DOCENTE: Prof.ssa SABRINA MANTACI

PREREQUISITI	Per poter comprendere i contenuti e per raggiungere gli obiettivi di apprendimento previsti è necessario che lo studente sia capace di elaborare un codice in un linguaggio di programmazione (in particolare il linguaggio C), che conosca elementi di Informatica Teorica (linguaggi, automi, espressioni regolari e grammatiche) e di Algoritmi (liste, pile, alberi, grafi, tabelle hash).
RISULTATI DI APPRENDIMENTO ATTESI	Conoscenza e capacita' di comprensione Il corso intende fornire agli studenti le nozioni necessarie per comprendere ed affrontare le diverse problematiche relative alle fasi della compilazione con particolare attenzione all'analisi lessicale, sintattica e semantica che trovano applicazione anche in altri contesti (traduzioni di linguaggi, parser, scanner). Il corso si prefigge anche di trasmettere la conoscenza di importanti strumenti di generazione automatica di parser e scanner. Capacita' di applicare conoscenza e comprensione Il corso si pone come obiettivo principale la comprensione, da parte dello studente, dei modelli computazionali e delle tecniche che gestiscono il funzionamento degli analizzatori lessicali e sintattici in un moderno compilatore, al fine di applicare tali metodologie da un lato per la generazione automatica o manuale di tali analizzatori, e dall'altro per la trasformazione e l'analisi dei testi guidata dalla sintassi. Gli studenti avranno modo di applicare le conoscenze acquisite anche attraverso un elaborato realizzato in gruppo facendo uso degli strumenti di generazione automatica di parser e scanner. Autonomia di giudizio Gli studenti saranno guidati ad apprendere in maniera critica e responsabile gli argomenti che verranno loro proposti in aula e in laboratorio. Ciascuno studente avra' inoltre occasione di arricchire la propria autonomia di giudizio attraverso la realizzazione di un progetto o di un elaborato che sara' parte integrante della prova di valutazione. Abilita' comunicative Attraverso le attivita' di laboratorio previste, il corso tendera' a sviluppare negli studenti l'interazione e la capacita' di saper lavorare in gruppo, di confrontarsi sulle problematiche al fine di individuare le soluzioni in base alle conoscenze acquisite durante il corso. L'acquisizione delle abilita' comunicative sara' realizzata tramite la partecipazione attiva dello studente alle attivita' di laboratorio nonche' l'esposizione dei risultati del lavoro individuale o di gruppo su argomenti o problematiche proposti dal docente. Capacita' d'apprendimento Il materiale proposto in aula e in laboratorio permetteranno di sviluppare le capacita' di apprendimento degli studenti, i quali saranno in grado di "interrogare" in modo integrato le proprie conoscenze-competenze a fronte delle problematiche affrontate. Gli studenti saranno stimolati inoltre ad una conoscenza piu approfondita e critica dei linguaggi di programmazione a loro gia' noti, tramite lo studio di come avviene il processo di compilazione di tali linguaggi. Competenze acquisite: Lo studente/la studentessa acquisirà la competenza di costruire autonomamente un analizzatore lessicale e un analizzatore sintattico, sia ai fini della compilazione, sia per altri fini, legati per esempio all'analisi automatica di un testo
VALUTAZIONE DELL'APPRENDIMENTO	E' prevista una prova in itinere che mira a verificare l'acquisizione delle conoscenze relative all'analisi lessicale e all'analisi sintattica. In particolare, la prova prevede 4 domande a risposta aperta, mirate a verificare la capacita' di costruzione di un determinato algoritmo di parsing per un dato linguaggio. La suddetta prova viene valutata in trentesimi (secondo la tabella descritta nel seguito) e mira a verificare le conoscenze acquisite, le capacita' elaborative e la proprieta' di linguaggio. Il punteggio massimo si ottiene se viene dimostrato un pieno possesso delle conoscenze e delle tecniche algoritmiche studiate e una spiccata capacita' di elaborare e applicare le conoscenze acquisite per risolvere i problemi proposti e un'ottima proprieta' di linguaggio. Alla fine del corso agli studenti (preferibilmente in gruppo) viene assegnato un progetto di studio in cui viene richiesto di progettare e implementare un analizzatore lessicale sintattico-semantico con l'ausilio di Flex e Bison. Tale prova mira a verificare la capacita' di saper applicare le conoscenze acquisite, ovvero la capacita' di definire un analizzatore lessicale, un analizzatore sintattico e le azioni semantiche per la risoluzione di un problema specifico. Tale prova inoltre mira a verificare la capacita' di interazione e di lavoro in gruppo. Al termine dell'elaborazione del progetto, gli studenti dovranno descrivere la loro soluzione dimostrando la loro capacita' espositiva. Tale attivita' viene valutata in trentesimi facendo riferimento alla tabella di valutazione descritta di seguito. La media delle valutazioni delle due prove diventa il voto di partenza per la prova

	orale, che prevede fino a tre domande che mirano a verificare l'acquisizione delle conoscenze relative agli argomenti trattati durante il corso. La valutazione finale e' espressa in trentesimi e tiene conto della tabella di valutazione descritta di seguito. Tabella di valutazione: Valutazione: Eccellente Voto: 30 - 30 e lode Esito: Ottima conoscenza degli argomenti, ottima proprieta' di linguaggio, buona capacita' analitica, efficace capacita' di interazione e di lavoro in gruppo, lo studente e' in grado di applicare le conoscenze per risolvere i problemi proposti. Valutazione: Molto buono Voto: 27-29 Esito: Buona padronanza degli argomenti, piena proprieta' di linguaggio, buona capacita' di interazione e di lavoro in gruppo, lo studente e' in grado di applicare le conoscenze per risolvere i problemi proposti. Valutazione: Buono Voto: 24-26 Esito: Conoscenza di base dei principali argomenti, discreta proprieta' di linguaggio, discreta capacita' di lavoro in gruppo, con limitata capacita' di applicare in modo autonomo le conoscenze alla soluzione dei problemi proposti. Valutazione: Soddisfacente Voto: 21-23 Esito: Non ha piena padronanza degli argomenti principali dell'insegnamento ma ne possiede le conoscenze, soddisfacente proprieta' di linguaggio, scarsa capacita' di saper applicare in modo autonomo le conoscenze acquisite. Valutazione: Sufficiente Voto: 18-20 Esito: Minima conoscenza di base degli argomenti principali dell'insegnamento e del linguaggio tecnico, scarsissima o nulla capacita' di saper applicare in modo autonomo le conoscenze acquisite. Valutazione:Insufficiente Esito: Non possiede alcuna conoscenza degli argomenti trattati nel corso
OBIETTIVI FORMATIVI	L'insegnamento ha due obiettivi. Il primo e' di natura culturale. Si vuole cioe' introdurre una teoria, quella alla base della realizzazione dei compilatori per i linguaggi di programmazione, che rappresenta probabilmente il piu' importante contributo scientifico allo sviluppo e alla diffusione delle tecnologie informatiche. Il secondo obiettivo e' piu' applicativo. Lo studente alla fine del corso sarà capace di costruire in maniera indipendente un analizzatore lessicale e un analizzatore sintattico, e svolgere le relative azioni semantiche, rendendolo abile a risolvere problemi di questa natura al livello industriale. Anche qualora lo studente non venga mai coinvolto (nella propria carriera lavorativa) nel progetto di un nuovo compilatore, e' comunque altamente probabile che ella/egli si trovi spesso ad utilizzare metodologie e tecniche che di tale teoria fanno parte: dagli automi e le grammatiche come formalismi per definire il comportamento di un sistema, alle tecniche per realizzare traduttori eventualmente piu' semplici di un compilatore vero e proprio (ad esempio, l'interprete di un file di configurazione). Verra' studiata la struttura generale di un compilatore, ponendo poi particolare enfasi agli aspetti di analisi lessicale e di parsing. Verranno studiati parser di tipo top-down (a discesa ricorsiva, predittivi e LL(1)) e di tipo bottom-up (SLR, LR e LALR). Infine, saranno esaminati esempi di traduttori per semplici linguaggi.
ORGANIZZAZIONE DELLA DIDATTICA	La didattica e' organizzata mediante lezioni frontali in aula e in laboratorio, in cui si prevede di realizzare sia un trasferimento della conoscenza ma anche, con il supporto del mezzo informatico, la contestuale verifica dell'acquisizione delle competenze pratiche richieste. Per incrementare il coinvolgimento degli studenti si utilizzano anche metodologie di lavoro di gruppo.
TESTI CONSIGLIATI	Testo principale (main textbook) A. Aho, M. Lam, R. Sethi, J. Ullman COMPILATORI. Principi, Tecniche e strumenti Addison Wesley (2009) Seconda edizione ISBN: 9788871925592 (ogni altra edizione del libro va bene ai fini del corso / any other edition of the book (even the one in english) is good for the porpouse of the course) Testi di consultazione (consultation textbooks) A. Aho, M. Lam, R. Sethi, J. Ullman Compilers, Principles, Techniques & Tools

	Addison Wesley Stefano Crespi Reghizzi Linguaggi Formali e Compilazione Esculapio (2015)

PROGRAMMA

ORE	Lezioni
4	Introduzione: Linguaggi di Programmazione e Processori di Linguaggi di Programmazione. Compilatori e Linguaggi. Linguaggi Macchina, Linguaggi Assembly ed evoluzione dei linguaggi di programmazione. Compilatori e Interpreti. Macchina Virtuale. Struttura di un compilatore: Preprocessore. Linker e Loader. Fasi della compilazione. Front End e Back End. Passate di un compilatore.
6	Analisi lessicale: Operazioni preliminari. Token e Lessemi. Errori lessicali. Token e espressioni regolari. Definizioni regolari. Eliminazioni di ambiguita. Automi a stati finiti. Implementazioni di DFA. Simulazioni di NFA. Generazione automatica di scanner. Uso di Flex.
6	Risoluzione di esercizi sull'analisi lessicale con l'ausilio del software Flex.
6	Analisi sintattica: Grammatiche context-free. Alberi di derivazione. Grammatiche ambigue. Automi a pila deterministici e non deterministici. Complessita' di calcolo di un automa a pila. Algoritmo di Earley. Errori sintattici e metodi di gestione degli errori.
6	Parser discendenti o top-down: Parser a discesa ricorsiva. Parser LL(1). Eliminazione della ricorsione sinistra. Fattorizzazione sinistra. Insiemi First e Follow. Parser ascendenti o bottom-up: Parser shift-reduce. Parser LR(0). Parser SLR. Parser LR(1). Parser LALR(1). Proprieta' dei linguaggi e delle grammatiche LR(k). Confronto tra le grammatiche LL(k) e LR(k). Generatori automatici di parser. Uso di Bison.
6	Analisi semantica: Semantica statica e dinamica. Grammatiche con attributi. Semantica guidata dalla sintassi. Albero sintattico decorato. Calcolo degli attributi. Grafo delle dipendenze. Grammatiche con S- attributi. Grammatiche con L-attributi. Ordinamento topologico del grafo delle dipendenze. Tabella dei simboli. Vari tipi di implementazioni tramite array, liste concatenate e ABR. Implementazione tramite hash table with chaining. Attributi di visibilita' e metodi di realizzazione. Type checking. Equivalenza di tipi. Type coercion.
6	Risoluzione di esercizi sull'analisi sintattica con l’ausilio del software Bison.
6	L'uso della tabella dei simboli
2	Generazione del codice: Codice intermedio. Codice a tre indirizzi. Strutture dati per l’implementazione del 3AC. Codice per macchina virtuale. P-code. Ottimizzazione del codice. Esempi di ottimizzazioni indipendenti dalla macchina. Generatori di codice oggetto. Esempi di ottimizzazioni dipendenti dalla macchina.