UNIVERSITÀ DEGLI STUDI DI PALERMO

DIPARTIMENTO	Ingegneria
ANNO ACCADEMICO OFFERTA	2022/2023
ANNO ACCADEMICO EROGAZIONE	2022/2023
CORSO DILAUREA MAGISTRALE	INGEGNERIA INFORMATICA
INSEGNAMENTO	LINGUAGGI E TRADUTTORI
TIPO DI ATTIVITA'	B
AMBITO	50369-Ingegneria informatica
CODICE INSEGNAMENTO	04761
SETTORI SCIENTIFICO-DISCIPLINARI	ING-INF/05
DOCENTE RESPONSABILE	LO RE GIUSEPPE Professore Ordinario Univ. di PALERMO
ALTRI DOCENTI	
CFU	9
NUMERO DI ORE RISERVATE ALLO STUDIO PERSONALE	144
NUMERO DI ORE RISERVATE ALLA DIDATTICA ASSISTITA	81
PROPEDEUTICITA'	
MUTUAZIONI	
ANNO DI CORSO	1
PERIODO DELLE LEZIONI	2° semestre
MODALITA' DI FREQUENZA	Facoltativa
TIPO DI VALUTAZIONE	Voto in trentesimi
ORARIO DI RICEVIMENTO DEGLI STUDENTI	**LO RE GIUSEPPE** Martedì 15:00 17:00

DOCENTE: Prof. GIUSEPPE LO RE

PREREQUISITI	Buona conoscenza dei costrutti tipici dei linguaggi di programmazione procedurali e ad oggetti; conoscenza dell'organizzazione della memoria di un calcolatore e del funzionamento del sistema operativo.
RISULTATI DI APPRENDIMENTO ATTESI	Conoscenza e capacita' di comprensione (knowledge and understanding): Lo studente, al termine del corso, avra' acquisito conoscenze e metodologie attinenti alle problematiche relative alle diverse fasi della compilazione con particolare attenzione all'analisi lessicale, sintattica e semantica ma che trovano applicazione anche in altri contesti (traduzioni di linguaggi, parser, scanner). Il corso si prefigge anche di trasmettere la conoscenza dei piu' importanti strumenti di generazione automatica di parser e scanner. Conoscenza e capacita' di comprensione applicate (applying knowledge and understanding): Lo studente sara' in grado comprendere il funzionamento degli analizzatori lessicali e sintattici, gli strumenti pratici per la realizzazione di tali analizzatori, il procedimento richiesto per trasformare gli analizzatori in traduttori, alcuni aspetti avanzati della compilazione di linguaggi moderni ed alcune tecniche di analisi automatica di correttezza di programmi. Il corso si propone anche di invitare gli studenti ad una conoscenza piu' approfondita dei linguaggi di programmazione a loro gia' noti, tramite lo studio di come tali linguaggi possono essere compilati. Autonomia di giudizio (making judgements): Lo studente sara' in grado di seguire i trend moderni nell'ambito della progettazione di compilatori e traduttori; sara' in grado di raccogliere i dati necessari alla valutazione delle prestazioni di un particolare compilatore, e di interpretare i risultati della valutazione; infine, sara' in grado di elaborare i requisiti necessari alla progettazione di un nuovo compilatore, e di valutare l'efficacia di diverse soluzioni alternative. Abilita' comunicative (communication skills): Lo studente acquisira' la capacita' di comunicare ed esprimere problematiche inerenti all'oggetto del corso; sara' in grado di sostenere conversazioni su tematiche relative ai moderni linguaggi e tecniche di compilazione, di confrontare diversi metodi di compilazione, e di offrire possibili soluzioni. Capacita' di apprendere (learning skills): Lo studente avra' appreso i nessi tra le tematiche dei compilatori, della progettazione dei linguaggi, e dell'importanza di adeguate tecniche di ottimizzazione, e questo gli consentira' di proseguire gli studi ingegneristici con un elevato grado di autonomia.
VALUTAZIONE DELL'APPRENDIMENTO	La valutazione avverra' mediante somministrazione di una prova scritta, al superamento della quale si potra' accedere all'esame orale. La prova scritta vertera' sulla costruzione di parser tabellari e sulla progettazione di costrutti di un linguaggio assunto come caso di studio. L'esame orale consistera' in una discussione dello svolgimento della prova scritta e nella discussione sugli argomenti del corso e degli elaborati svolti durante le esercitazioni teoriche. La valutazione dell'apprendimento sara' focalizzata sulla valutazione dei risultati attesi indicati nella presente scheda, in accordo con i descrittori di Dublino. Il voto finale sara' espresso in trentesimi nell'intervallo 18/30 - 30/30 con lode. - Valutazione dell'obiettivo 1: Conoscenza e capacita' di comprensione La verifica di questo obiettivo consistera' in un esame orale riguardante la discussione degli argomenti del programma. L'obiettivo 1 contribuira' per il 30% al voto finale. - Valutazione dell'obiettivo 2: Capacita' di applicare conoscenza e comprensione La verifica di questo obiettivo si basera' sullo svolgimento della prova scritta. Per la verifica di questo obiettivo l'esame comprende la costruzione di parser tabellari e la traduzione di costrutti di un linguaggio assunto come caso di studio. L'obiettivo 2 contribuira' per il 35% al voto finale. - Valutazione dell'obiettivo 3: Autonomia di giudizio Per la verifica di questo obiettivo la prova scritta includera' almeno un esercizio sulla progettazione della sintassi e relativa traduzione di un costrutto di programmazione per un ipotetico linguaggio procedurale in in cui lo studente sia stimolato a effettuare delle scelte progettuali in autonomia. L'obiettivo 3 contribuira' per il 15% al voto finale. - Valutazione dell'obiettivo 4: Abilita' comunicative La verifica di questo obiettivo l'esame avverra' mediante esame orale sugli argomenti del corso con discussione della traduzione dei costrutti richiesta per la prova scritta. L'obiettivo 4 contribuira' per il 10% al voto finale. - Valutazione dell'obiettivo 5: Capacita' di apprendere La verifica di questo obiettivo comprende la discussione durante la prova orale di alcuni degli argomenti piu' avanzati trattati a lezione, il cui approfondimento e' lasciato agli studenti, in modo da verificare la capacita' di stabilire un nesso tra i concetti teorici e la loro pratica realizzazione. L'obiettivo 5 contribuira' per il 10% al voto finale.

OBIETTIVI FORMATIVI	L'obiettivo del corso e' quello di fornire gli strumenti fondamentali, sia formali sia pratici, per la definizione dei linguaggi di programmazione e dei loro compilatori. La prima parte del corso e' dedicata alla teoria dei linguaggi formali ed alla sua relazione con la teoria degli automi. A tale scopo, si enfatizzano gli aspetti generativi e riconoscitivi dei linguaggi formali. Nella seconda parte si illustrano gli aspetti piu' significativi dei linguaggi di programmazione, la loro evoluzione ed i concetti che stanno alla base della compilazione dei linguaggi di alto livello. Infine si presentano strumenti per la generazione automatica di analizzatori lessicali e sintattici, tanto bottom-up, come LEX e YACC, quanto top-down. Parte integrante del corso e' la progettazione e la realizzazione di strumenti per l'analisi di programmi scritti in un linguaggio procedurale.
ORGANIZZAZIONE DELLA DIDATTICA	Lezioni frontali con analisi e discussione in aula di casi di studio. Esercitazioni teoriche e pratiche su costrutti tipici dei linguaggi. Presentazioni e discussioni in aula di progetti e implementazioni.
TESTI CONSIGLIATI	Aho, Lam, Sethi, Ullman, “Compilers: Principles, Techniques and Tools”, 2nd edition, Addison Wesley, 2006. ISBN 0-321-49169-6

PROGRAMMA

ORE	Lezioni
3	Introduzione al corso. Descrizione della struttura di un compilatore: fasi di analisi e di sintesi. L'evoluzione dei linguaggi di programmazione e la scienza dei traduttori automatici, applicazione della tecnologia dei compilatori. Elementi di base di un linguaggio di programmazione.
8	L'analisi lessicale. Confronto tra analisi lessicale e parsing. Token, pattern e lessemi. Gestione dei tipici errori lessicali. Specifiche dei token. Definizione formale di espressione regolare. Principali proprieta' delle espressioni regolari.
8	Automi a stati finiti; automi deterministici e non deterministici. Proprieta' di NFA e DFA e rappresentazione mediante tabelle. Conversione tra le diverse rappresentazioni mediante NFA, DFA ed espressioni regolari. Ottimizzazione dei lexer basati su DFA.
4	Le proprieta' di chiusura dei linguaggi regolari. Il pumping lemma per linguaggi regolari.
6	Analisi Sintattica: Il ruolo del parser. Grammatiche libere dal contesto. Gestione e ripristino da errori. Derivazioni ed alberi di parsing. Ambiguita, verifica di un linguaggio generato da una grammatica, confronto tra grammatiche libere dal contesto.
6	Top-down parsing and LL(1) grammars. Recursive descent parser. Nonrecursive Predictive Parsing. Bottom-up syntax analysis. LR(1) grammars. Shift-reduce parsing. LR(0) items. Closure and GOTO functions. Bottom-up SLR parser. Canonical LR parser. LALR parser. Efficient implementation of bottom-up parsers.
6	Traduzione guidata dalla sintassi. SDD. Uso di attributi sintetizzati ed ereditati. SDT. Implementazione di SDT per i diversi tipi di parser. Generazione di codice on-the-fly. Problematiche relative al parsing ascendente con SDD L- attributed.
6	Codice intermedio. Rappresentazione delle istruzioni mediante grafi, e codice a 3 indirizzi. Rappresentazione a quadruple e a triple.
6	Utilizzo dei tipi; traduzione della definizione di tipi primitivi e type checking. Traduzione di istruzioni di controllo di flusso. Traduzione di dichiarazione e uso di variabili di tipi composti (array e record). Compilazione one-pass e tecnica di backpatching
6	Ambienti a tempo di Esecuzione: Organizzazione della Memoria, Allocazione dello Spazio su Pila, Accesso ai Dati non locali sulla Pila, Gestione dello Heap, Introduzione al Garbage Collection,

ORE	Esercitazioni
6	Uso del generatore di analizzatori lessicali flex. Implementazione delle espressioni regolari con il generatore di lexer flex. Uso in bash, egrep.
4	Dimostrazioni per induzione sulla correttezza, ambiguita. Operazioni sulle grammatiche. Disambiguazione di grammatiche, left-factoring, eliminazione della left-recursion.
12	Esercizi sulla costruzione di tabelle di parsing top-down; dimostrazioni per induzione sulle grammatiche. Esercizi sulla costruzione di tabelle di parsing bottom-up. Utilizzo del generatore automatico di parser yacc/ bison.