UNIVERSITÀ DEGLI STUDI DI PALERMO

DIPARTIMENTO	Matematica e Informatica		
ANNO ACCADEMICO OFFERTA	2023/2024		
ANNO ACCADEMICO EROGAZIONE	2023/2024		
CORSO DILAUREA	INFORMATICA		
INSEGNAMENTO	PROGRAMMAZIONE E LABORATORIO C.I.		
CODICE INSEGNAMENTO	05880		
MODULI	Si		
NUMERO DI MODULI	2		
SETTORI SCIENTIFICO-DISCIPLINARI	INF/01		
DOCENTE RESPONSABILE	LO BOSCO GIOSUE'	Professore Associato	Univ. di PALERMO
ALTRI DOCENTI	LO BOSCO GIOSUE'	Professore Associato	Univ. di PALERMO
	ROMANA GIUSEPPE	Professore a contratto	Univ. di PALERMO
CFU	12		
PROPEDEUTICITA'			
MUTUAZIONI			
ANNO DI CORSO	1		
PERIODO DELLE LEZIONI	Annuale		
MODALITA' DI FREQUENZA	Facoltativa		
TIPO DI VALUTAZIONE	Voto in trentesimi		
ORARIO DI RICEVIMENTO DEGLI STUDENTI	**LO BOSCO GIOSUE'** Martedì 15:00 17:00 Ufficio al secondo piano del Dipartimento di Matematica e Informatica, Stanza 203. E' suggerita la prenotazione		

DOCENTE: Prof. GIOSUE' LO BOSCO

PREREQUISITI	Nessun prerequisito.
RISULTATI DI APPRENDIMENTO ATTESI	Conoscenza e capacità di comprensione: acquisizione dei concetti fondamentali della programmazione strutturata; strutture dati elementari statiche e dinamiche; semplici algoritmi fondamentali di ricerca e di ordinamento; definizione ricorsiva di soluzioni; padronanza dei costrutti fondamentali del linguaggio di programmazione C. Capacità di applicare conoscenza e comprensione: capacità di affrontare semplici problemi computazionali mediante scelta di strutture dati e algoritmi appropriati; capacità di programmazione in linguaggio di programmazione C; capacità di validare, mediante la scrittura di semplici programmi, le nozioni teoriche; capacità di comprensione degli errori rilevati in fase di compilazione ed esecuzione di semplici programmi scritti in C; capacità di decomporre problemi complessi in problemi più semplici da un punto di vista computazionale. Autonomia di giudizio: saper individuare le modalità più appropriate nel passaggio dei parametri; saper confrontare due semplici programmi in termini di efficienza di calcolo e invarianza rispetto ai cambiamenti; saper individuare una soluzione algoritmica efficiente di problemi semplici. Abilità comunicative: proprietà di espressione nella presentazione delle nozioni di base dei linguaggi di programmazione e della programmazione imperativa. Capacità d'apprendimento: sapere affrontare lo studio di diversi linguaggi di programmazione e sapere contestualizzare le abilità acquisite in problemi concreti nell'ambito lavorativo. Competenze acquisite. Saper . creare algoritmi per illustrare gli approcci iterativo, ricorsivo e divide-et-impera alla soluzione di un problema e usare un linguaggio di programmazione per realizzare, verificare e correggere le soluzioni; . decomporre un programma per un cliente mediante identificazione delle componenti di dato e di funzione di tipi di dato astratti e implementare un tipo di dato astratto con accoppiamento di dati e funzionalità; . progettare, realizzare, verificare e correggere un programma che usa i costrutti fondamentali della programmazione e del calcolo elementare, inclusi l'input/output, le strutture condizionali e iterative, la definizione di funzioni, il passaggio dei parametri; . presentare i costi e i benefici delle realizzazioni di strutture dati statiche e dinamiche, scegliendo la struttura più adeguata per un dato problema; . dimostrare gli errori tipici di programmazione, costruire e correggere programmi usando le librerie standard disponibili per un dato linguaggio.
VALUTAZIONE DELL'APPRENDIMENTO	L'esame è costituito da una prova scritta, da una prova pratica e da un breve colloquio orale. La prova scritta è suddivisa in due parti, ciascuna corrispondente ad uno dei due moduli del corso. Essa è costituita da dieci domande a risposte multiple e da due esercizi di programmazione. A ciascuna risposta vengono assegnati 2 punti (risposta esatta), 0 punti (nessuna risposta), o -0.5 punti (risposta sbagliata). A ciascun esercizio di programmazione viene assegnato un massimo di 6 punti. La prova scritta si considera superata solo se il suo punteggio risulta essere superiore a 15. La prova pratica consiste nello sviluppo di un semplice programma inerente la gestione di liste, array o file. La verifica della prova pratica consiste nell'esecuzione del programma e nell'analisi del codice, ed è seguita da un breve colloquio con ulteriori domande volte a valutare la conoscenza delle principali nozioni di programmazione. La prova pratica/orale è complessivamente valutata in trentesimi. Il voto finale tiene conto dei voti conseguiti nella prova scritta e nella prova pratica/orale. Le fasce di punteggio corrispondono ai giudizi qualitativi seguenti: 18-20: la conoscenza della materia e le competenze di programmazione sono sufficienti; 21-23: la conoscenza della materia e le competenze di programmazione sono discreti; 24-25: la conoscenza della materia e le competenze di programmazione sono buoni; 26-27: la conoscenza della materia e le competenze di programmazione sono molto buoni; 28-30: la conoscenza della materia e le competenze di programmazione sono ottimi. La lode è riservata agli studenti che dimostrano nella prova pratica e nella prova orale eccellenti competenze di programmazione e un'ottima padronanza della materia. E' prevista una prova scritta in itinere alla fine del primo modulo. La forma della

	prova in itinere è simile alle prove scritte delle sessioni d’esame, e il superamento della prova in itinere comporta su richiesta dello studente l’esonero dalla prima parte della prova scritta.
ORGANIZZAZIONE DELLA DIDATTICA	Lezioni frontali in aula e in laboratorio

MODULO
PROGRAMMAZIONE STRUTTURATA IN C

Prof. GIOSUE' LO BOSCO

TESTI CONSIGLIATI

P.J. Deitel and H.M. Deitel. Il linguaggio C: Fondamenti e tecniche di programmazione, 9/Ed. Pearson, ISBN 9788891901651, 2016. (9/Ed in uscita a Settembre 2022)

TIPO DI ATTIVITA'	A
AMBITO	50168-Formazione informatica di base
NUMERO DI ORE RISERVATE ALLO STUDIO PERSONALE	86
NUMERO DI ORE RISERVATE ALLE ATTIVITA' DIDATTICHE ASSISTITE	64

OBIETTIVI FORMATIVI DEL MODULO

Il modulo si propone di fornire allo studente gli strumenti teorici e pratici per la progettazione di un programma al computer nei suoi aspetti fondamentali: la rappresentazione dei dati in strutture e la formulazione di semplici algoritmi che fanno uso delle fondamentali strutture di controllo, selezione e iterazione. Il linguaggio di programmazione utilizzato e' il C, per la sua diffusione e per essere propedeutico rispetto alla maggior parte dei moderni linguaggi di programmazione. Un terzo delle ore di lezione frontale previste viene svolto in aula attrezzata con computer, in maniera che gli studenti possano svolgere direttamente gli esempi e gli esercizi proposti dal docente. Questa modalita' didattica consente il trasferimento delle competenze teoriche in esempi pratici di implementazione.

PROGRAMMA

ORE	Lezioni
4	Introduzione al corso di Programmazione. Architettura di un elaboratore secondo Von Neumann. Rappresentazione dell'informazione. Rappresentazione in binario di interi, interi relativi, reali, caratteri. Il codice ASCII. Le stringhe.
4	Cambiamento di base per la rappresentazione degli interi. Definizione di Algoritmo. Esempi di algoritmi. I diagrammi di flusso, rappresentazione di un algoritmo attraverso un diagramma di flusso. Cenni sulla complessita' computazionale di un algoritmo.
4	Il linguaggio C. Struttura di un programma in C. Compilazione, linkng, preprocessor. Librerie fondamentali. Identificatori. Le costanti e le variabili. Dichiarazione e assegnazione. Il tipo intero, float e double. Il tipo char. Funzioni di input/output base.
4	Gli operatori in C: aritmetici, relazionali, logici, bit a bit. Ordine di priorita' degli operatori. I costrutti di selezione in C: If, then, else e switch, case. Macro. Compilazione condizionale.
3	I costrutti di iterazione in C. Costrutto "for". I costrutti di iterazione while, do..while. Equivalenza dei costrutti di iterazione.
3	Array in C. Dichiarazione statica di un array. Stringhe in C come array statici di caratteri.
4	Puntatori. Aritmetica dei puntatori. Dichiarazione dinamica di un array. Stack e Heap. Stringhe in C come array dinamici di caratteri.
4	Array a piu' dimensioni. Dichiarazione statica e dinamica di un array multidimensionale. Equivalenza tra strutture multidimensionali e monodimensionali e loro rappresentazione in memoria. Le funzioni in C. La dichiarazione, la definizione e la chiamata di funzioni. Passaggio dei parametri per valore e per indirizzo.
2	I file in C. File binari e testuali. Le funzioni per la lettura e scrittura in un file. Accesso casuale e sequenziale.

ORE	Laboratori
4	Codifica in linguaggio C di primi semplici programmi con i costrutti di selezione. Compilazione e linking con GCC. Uso del comando make e dei makefile .
4	Implementazione in C di programmi che usano costrutti iterativi.
4	Implementazione in C di programmi per l’inserimento e la visualizzazione di un array statico. Implementazione in C di programmi per trovare la lunghezza di una stringa, per il confronto tra due stringhe, per la ricerca di una sotto stringa.
4	Implementazione in C di programmi per l’inserimento e la visualizzazione di un array dinamico. Implementazione in C di programmi per trovare la lunghezza di una stringa, per il confronto tra due stringe, per la ricerca di una sotto stringa utilizzando i puntatori a carattere.
6	Implementazione in C di programmi per l’inserimento e la visualizzazione di array statici e dinamici a piu' dimensioni. Implementazione in C del calcolo del prodotto tra due matrici, sia tramite array statico che dinamico.
6	Implementazione in linguaggio C degli algoritmi di ordinamento, in particolare Insertion e Selection Sort. Implementazione in linguaggio C.della ricerca di un elemento in array non ordinati e ordinati.
4	Implementazione in linguaggio C di un programma per leggere e scrivere su un file testuale o binario.

MODULO
STRUTTURE DATI ASTRATTE

Prof. GIUSEPPE ROMANA

TESTI CONSIGLIATI

P.J. Deitel and H.M. Deitel. Il linguaggio C: Fondamenti e tecniche di programmazione, 9/Ed. Pearson, September 2022.
R. Sedgewick. Algoritmi in C, 4/Ed. Pearson, ISBN 9788891900746, 2015. (strutture astratte di dati / abstract data structures)

TIPO DI ATTIVITA'	A
AMBITO	50168-Formazione informatica di base
NUMERO DI ORE RISERVATE ALLO STUDIO PERSONALE	86
NUMERO DI ORE RISERVATE ALLE ATTIVITA' DIDATTICHE ASSISTITE	64

OBIETTIVI FORMATIVI DEL MODULO

Il modulo si propone di fornire allo studente gli strumenti teorici e pratici per la progettazione di strategie ricorsive e iterative per la risoluzione di problemi. Vengono affrontate la gestione dinamica della memoria e la costruzione di strutture dati dinamiche, alcune delle quali definite mediante l'ausilio dei puntatori. Si implementano, in diverse versioni e con diverso grado di complessità, funzioni ricorsive ed iterative per la costruzione e per la gestione delle strutture dati, consentendo così allo studente di acquisire familiarità con la programmazione, gli algoritmi, e la complessità computazionale. Viene affrontata ed esemplificata la costruzione di strutture dati, a crescente livello di astrazione.
Un metà delle ore di lezione frontale previste viene svolta in aula attrezzata con calcolatori, in maniera che gli studenti possano svolgere direttamente gli esempi e gli esercizi proposti dal docente. Questa modalità didattica consente una efficace trasmissione di competenze teorico-pratiche.

PROGRAMMA

ORE	Lezioni
4	La ricorsione. Esempi di funzioni ricorsive: il fattoriale, la somma di una successione di interi, i numeri di Fibonacci. Confronto tra iterazione e ricorsione. Ricorsione in coda. Introduzione alla programmazione dinamica. Funzioni ricorsive su array: ricerca binaria ricorsiva e algoritmo di ordinamento mergesort. Confronto computazionale del mergesort con gli algoritmi di ordinamento elementari.
4	Esempi di strategie ricorsive per la soluzione di problemi su array e su stringhe. Debugging e profiling di programmi C. Puntatori e oggetti dinamici. Allocazione e deallocazione di memoria. Principio del minimo privilegio. Puntatori ed array.
4	Puntatori a funzione. Esempi: sorting bidirezionale, menu. Le strutture e i tipi derivati. Operazioni sulle strutture. Strutture e puntatori. Passaggio di argomenti per valore e per riferimento. Embedding di array. Strutture e funzioni. Programmazione dinamica: il problema dello zaino 0-1, cammino più redditizio da NW a SE in una matrice. Unioni e condivisione di spazio.
4	Nodi per strutture dinamiche. Liste concatenate. Esempio: lista ordinata di caratteri, inserimento e cancellazione. Confronto tra array e liste. Esempi: inversione ricorsiva di una lista, ricerca in una lista. Fusione di liste ordinate. Mergesort di liste concatenate. Mergesort dal basso su array. Mergesort dal basso su liste. Mergesort naturale.
4	Struttura dati Pila e sua realizzazione con lista concatenata. Struttura dati Coda e sua realizzazione con lista concatenata. Tipo astratto di dati (ADT): necessità, caratteristiche, definizione. Il concetto di astrazione di tipo e la sua realizzazione. La pila come ADT: realizzazioni con lista concatenata, array, o array ridimensionabile (reallocazione dinamica). Nascondere i dati con l'attributo static.
4	La coda come ADT: realizzazioni con lista concatenata, array, o array ridimensionabile. Buffer circolare e ridimensionamento del buffer. Mergesort dal basso con coda di liste. Valutazione di espressioni postfisse mediante pila. Compilazione da più file sorgente e uso del pre-processore. Librerie statiche.
4	Operazioni orientate ai bit. Campi di bit. Costanti di enumerazione e tipi enumerati. Esempio: estrazione e rappresentazione dei contenuti dello zaino. Alberi e ADT albero binario di ricerca (BST). Inserimento in un BST. Visite di un albero. Ricerca in un BST. Stampa di un albero. Cancellazione in un BST. Modelli come gerarchie di parti, esempio: rappresentazione di un umanoide come albero.
4	File binari ad accesso casuale. ADT per code basato su file binari ad accesso casuale. Tipi di dato astratto di prima categoria (che supportano istanze multiple). Esempio: ADT numero complesso. La coda come ADT di prima categoria. Sistemi di code. La pila come ADT di prima categoria. Esempio: valutatore di espressioni infisse sui numeri complessi. ADT polinomio.

ORE	Laboratori
4	Esempi su funzioni ricorsive e iterative. Misura del tempo di esecuzione con time.h. Profiling con gprof. Trasformazione di ricorsione in iterazione. Massimo comun divisore in forma ricorsiva.

4	Mergesort ricorsivo e versione con switch a insertion sort per taglia piccola. Misura di performance. Regolazione della taglia di commutazione. Mergesort su file separato da quello di test, che funziona su dati di tipo qualsiasi, con array di void *. Esercizi: liste ordinate di caratteri; stampa ricorsiva invertita di lista; ricerca ricorsiva di un elemento in una lista; fusione di liste ordinate.
4	Esercizi: mazzo di carte; problema dello zaino ricorsivo; problema dello zaino ricorsivo con programmazione dinamica; problema dello zaino iterativo dal basso. Uso di valgrind per l'identificazione di violazioni di memoria. Introduzione allo strumento di debugging gdb.
4	Realizzazione di stack con liste. Modelli come gerarchie di parti, esempio: braccio meccanico in OpenGL. Realizzazione di queue con liste. Mergesort top-down con liste. Mergesort naturale bottom-up con liste e array ausiliario. Misure di prestazione.
4	Stack con liste come ADT. Stack con array come ADT. Stack con array con resizing. Stack con array con realloc. Stack con verifica di presenza. Astrazione dell'input-output dal tipo di dato base.
4	Queue ADT con liste, array e resizable array. Circular buffer. Mergesort su lista con coda ausiliaria, verifica di stabilità. Valutatore di espressioni postfisse.
4	Operazioni sui bit. Stampa dei bit di un unsigned int. Esempi di operazioni logiche e di shifting sui bit. Esempio: restituzione del contenuto nel problema dello zaino. Campi di bit. Costanti di enumerazione. Esercizio: lista di sottoliste.
4	Alberi binari e BST. BST ADT. Visite: in-ordine, pre-ordine, post-ordine. Esercizio: funzione di ricerca sul BST, complessità e misura di prestazione. Esercizio: stampa di un BST. Esercizio: Visita per livelli e realizzazione con Queue ADT. Cancellazione in un BST. Esempi: umanoide come albero e rendering OpenGL come visita in pre-ordine; rappresentazione e visualizzazione di un grafo. ADT polinomio. Valutazione di espressioni infisse sui numeri complessi.