IN110 Informatica 1 (Fondamenti)

A.A. 2009/2010

Prof. Marco Liverani

Algoritmi, strutture dati e programmazione in linguaggio C

1. Problemi ed algoritmi

Introduzione alle caratteristiche del calcolatore ed al rapporto programmatore/esecutore; compiti ed abilità del programmatore; principali caratteristiche ed abilità dell'esecutore, operazioni di base (logiche, aritmetiche e di confronto).

Modelli di macchina calcolatrice: la macchina RAM, cenni sulla macchina di Turing.

Linguaggi di programmazione: linguaggi imperativi e dichiarativi. Istruzioni fondamentali di un linguaggio di programmazione procedurale generico. Algoritmi e programmi; diagrammi di flusso. Regole della programmazione strutturata, cenni sul teorema di Jacopini-Böhm; approccio *top–down* alla soluzione di un problema.

2. Utilizzo del calcolatore

Architettura di un computer: CPU, memoria RAM, memoria di massa, bus, interfacce, dispositivi di input/output, rete. Il sistema operativo UNIX/Linux, principali comandi della *shell* del sistema operativo, connessione da remoto in emulazione di terminale, protocollo SSH. Utilizzo di editor di testo, compilazione ed esecuzione di programmi. GNUplot, utilizzo del programma per la visualizzazione di grafici di funzione. Latex, utilizzo del programma per la redazione di documenti di carattere scientifico.

3. Il linguaggio C

Organizzazione della memoria di un calcolatore, indirizzi, parole, puntatori. Codifica binaria. Tipi di dato, strutture dati (*array*, matrici, pile, code, code di priorità, liste, alberi, grafi). Linguaggio macchina, linguaggi di alto livello; compilatori ed interpreti, compilazione ed esecuzione di un programma C in ambiente UNIX/Linux. Il linguaggio C: scopi e principali caratteristiche.

La struttura di un programma C, l'inclusione degli *header*, dichiarazione delle variabili; le librerie. Tipi di dato elementari in linguaggio C: interi, *floating point*, *double*, *char*. Puntatori; aritmetica sui puntatori. *Array* e matrici e loro rappresentazione in memoria. Strutture dati complesse: liste, alberi, grafi; l'istruzione "`struct`". Operatore di assegnazione, operatori aritmetici in C in forma estesa e compatta. Strutture di controllo: "`if` ... `else` ...", "`while` ...", "`do` ... `while`", "`for` ...".

Funzioni: funzioni di libreria e funzioni definite dall'utente. Passaggio di parametri per valore e per indirizzo alle funzioni. Funzioni ricorsive. Funzioni di input/output: `printf`, `scanf`, `fprintf`, `fscanf`; funzioni per la gestione della memoria: `malloc`, `free`, `sizeof`; gestione di liste di record collegati tramite puntatori.

4. Algoritmi di ordinamento

Algoritmi di ordinamento elementari: *Insertion sort*, *Selection sort*, *Bubble sort*; l'approccio "divide et impera", l'algoritmo *Quick sort*. Strutture di tipo LIFO (*Last In First Out*), le pile; strutture di tipo FIFO (*First In First Out*), le code; code di priorità, gli *heap*. Algoritmi ottimi per l'ordinamento: *Heap sort*, *Merge sort*.

Complessità di un algoritmo, la notazione "O grande", analisi della complessità degli algoritmi di ordinamento.

5. Algoritmi elementari sui grafi

Definizioni principali: grafo, grafo orientato, grafo connesso; sottografo, sottografo indotto; cammino, cammino semplice, ciclo, grafo aciclico; alberi, foreste, *spanning tree* di un grafo. Strutture dati per la rappresentazione di grafi mediante un calcolatore: liste di adiacenza e matrici di adiacenza. Algoritmi di visita di un grafo: visita in ampiezza (BFS), visita in profondità (DFS), ordinamento topologico di un grafo. Problemi di cammino minimo su un grafo, l'algoritmo di Dijkstra.

Analisi della complessità degli algoritmi presentati. Cenni sulle classi dei problemi P, NP, NP-completi. Il problema "P=NP".

Testi consigliati

[1] A. Bellini, A. Guidi, *Linguaggio C - Guida alla programmazione*. McGraw-Hill (Quarta edizione, 2009).
[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduzione agli algoritmi e strutture dati*. McGraw–Hill (2005).
[3] M. Liverani, *Programmare in C*. Esculapio (2001).

Bibliografia supplementare

[4] P. Crescenzi, G. Gambosi, R. Grossi, *Strutture dati e algoritmi*. Pearson - Addison Wesley (2006).
[5] H.M. Deitel, P.J. Deitel, *C, corso completo di programmazione*. Apogeo (Seconda edizione, 2004).
[6] A. Kelley, I. Pohl, *C, didattica e programmazione*. Pearson - Addison Wesley (Quarta edizione, 2004).
[7] M. Liverani, *Qual è il problema? Metodi, strategie risolutive, algoritmi*. Mimesis (2005).
[8] M. Liverani, *Appunti su alcuni argomenti ed esercizi svolti, disponibili sul sito web del corso IN110: http://www.mat.uniroma3.it/users/liverani/IN110*.

Modalità d'esame

- valutazione in itinere ("esoneri")		■ SI	□ NO
- esame finale	scritto	■ SI	□ NO
	orale	■ SI	□ NO
- altre prove di valutazione del profitto (meglio descritte sotto)		□ SI	■ NO

L'esame si articola su due fasi successive: una prova scritta e una prova orale. Per accedere alla prova orale è necessario superare con un voto sufficiente la prova scritta, o le due prove di esonero previste. La prova scritta consiste nella soluzione di due esercizi: viene richiesto di presentare una codifica in linguaggio C di un algoritmo che risolva il problema dato. La prova orale consiste nella discussione di alcuni tra i principali argomenti trattati durante il corso (algoritmi di ordinamento, algoritmi di visita di grafi, strategia risolutiva, analisi della complessità, esempi, diagrammi di flusso; aspetti importanti del linguaggio C: allocazione della memoria, puntatori, ecc.).