E-10: Preparazione all'esame

Fondamenti di Informatica e Laboratorio T-AB

Corso di Laurea in Ingegneria dell'Automazione

Università di Bologna, A.A. 2008/2009

Paolo Torroni, Marco Montali

29 Maggio 2009

Struttura dell'esame

Le modalità d'esame sono descritte in dettaglio nel sito web del corso.

La valutazione comprende un'unica prova della durata di 4 ore, da svolgere in laboratorio, composta da:

A) una parte di teoria, per verificare la conoscenza dei principi di base e degli argomenti teorici affrontati a lezione;

B) una parte di analisi, per valutare le capacità di analisi di piccoli algoritmi implementati in linguaggio C;

C) una parte di Unix, per valutare la capacità di realizzare un semplice script per la shell di Unix;

D) una prova di programmazione in C, che consiste nella scrittura di un programma che risolva un problema dato, a partire da alcune specifiche.

Per raggiungere la sufficienza, bisogna ottenere una valutazione sufficiente **in ogni parte** (A—D).

Facoltativamente, e solo se si è già ottenuta la sufficienza alla prova in laboratorio, è possibile sostenere una prova orale, che verterà principalmente sugli aspetti in cui le altre prove hanno evidenziato lacune, e in generale su tutto il programma.

In aggiunta, sempre facoltativamente, è possibile presentare un progetto da discutere in sede di orale, su un argomento da concordarsi preliminarmente con i docenti.

Il massimo dei punti ottenibili in laboratorio è 30, il massimo dei punti all'orale è 2 senza progetto, 4 con progetto. L'orale può anche risultare in un voto negativo fino a -2 (quindi i voti vanno da -2 a +4).

Il voto complessivo che sarà verbalizzato è dato dalla somma dei punteggi delle varie parti. Nel caso di punteggio pari o superiore a 32, verrà verbalizzato 30 e lode.

Progetti

Formato del progetto

Il progetto andrà consegnato sotto forma di un file compresso (archivio, es. zip).

L'archivio deve espandersi in una cartella chiamata con CognomeNome del candidato.

All'interno della cartella si devono trovare:

- un file **pdf** (massimo 5 pagine) con la descrizione del problema, degli obiettivi, e della soluzione. Questa deve consistere in: struttura del progetto, prototipi delle funzioni, progetto dei dati, gestione errori, guida utente.
- un file **pdf** in cui si dichiara che il progetto è originale ed è stato svolto individualmente dal candidato o dai due candidati del team.
- i file sorgenti (**.c**, **.h**)
- gli altri eventuali file (es, i file di input)

Esempi di progetti

Un progetto può sviluppare un argomento visto nel corso, in modo più approfondito:

- Analisi comparativa sperimentale di nuovi algoritmi di ricerca o ordinamento, oltre a quelli visti a lezione.
- Definizione di un nuovo tipo di dato astratto e sua implementazione mediante libreria di funzioni. Alcuni esempi di dati astratti:
  - Alberi binari;
  - Code FIFO;
  - Code con priorità;
  - Intero a 256 bit;
  - Nuovi tipi inventati dal candidato (devono essere motivati da una possibile applicazione).

Un'altra possibilità è quella di usare il C per realizzare un interprete di un linguaggio di programmazione di base (3/4 istruzioni) in un dominio tra quelli visti nel corso:

- Simulatore di una CPU di base con un linguaggio assembly proprietario.
- Interprete di comandi per softbot (tipo Karel).

È anche possibile sviluppare argomenti collegati al corso di Fondamenti, ma non visti a lezione, come ad esempio Internet (XML, HTML, Wiki), text processing (LaTeX), sistemi operativi, intelligenza artificiale, o sistemi informativi:

- Gestione dati numerici a partire da file di testo strutturati (es. xml).
- Produzione di documenti Web (es. HTML o Wiki) o LaTeX a partire da file di input formattati.
- Comunicazione tra processi in Unix (problema dei 5 filosofi).
- Algoritmi con backtracking (e.g. N-regine).
- Progettazione di una "mini" base di dati con schema ER (2/3 tabelle e 2/3 query SQL).

In ogni modo, il docente è a disposizione per definire gli obiettivi del singolo progetto, e valutarne l'impegno assieme al candidato.

Modalità di presentazione del progetto

I progetti possono essere individuali o a coppia. Non sono consentiti team con più di due candidati.

È necessario **discutere la proposta del progetto con il docente** (su appuntamento: dopo la sessione di laboratorio o a ricevimento).

La **consegna del progetto** (per e-mail al docente: firmare le e-mail con nome, cognome e numero di matricola) deve avvenire **almeno 4 giorni lavorativi** prima della data fissata per l'orale.

La **discussione del progetto** in sede di orale è **obbligatoria**.

Il voto attribuito al progetto, comprensivo di esame orale, è fino a 4 punti, che si andranno a sommare ai punti ottenuti allo scritto.

È consentita la discussione del progetto solo al conseguimento di un voto positivo (≥ 18) allo scritto.

Domande per l'orale

Le domande che seguono sono solo esempi, e non sono in alcun modo esaustive. Ciascuna domanda può valere da 1 a 4 punti.

- Si definisca che cos'è un algoritmo: quali sono le caratteristiche fondamentali e quali sono alcune delle altre possibili caratteristiche.
- Che cosa definisce la sintassi di un linguaggio di programmazione? A cosa serve?
- Qual è la differenza tra sintassi e semantica di un linguaggio di programmazione?
- Cos'è una grammatica? come può essere definita in modo formale?
- Si definisca una BNF per il linguaggio le cui frasi corrette consistono di parole di 4 o 5 caratteri, in cui il primo carattere è uguale all'ultimo, e il vocabolario è costituito dai soli simboli `A` e `B`.
- Che cos'è un errore logico? Quali sono altri tipi di errori di programmazione? A che cosa serve il debugger?
- Qual è la differenza tra compilazione e interpretazione? Il linguaggio dei comandi di Unix è interpretato o compilato? Qual è il processo che parte da un insieme di file sorgenti C e arriva all'esecuzione di istruzioni da parte del calcolatore?
- Cosa sono le metodologie top-down e bottom-up?
- Quali sono le differenze tra funzione e procedura? Cos'è il modello cliente-servitore? Quali sono i meccanismi per il passaggio di informazione tra cliente e servitore? Nel caso delle procedure?
- Si espoga cosa sono parametri effettivi e parametri formali, il concetto di *binding* ed *environment*.
- Che cos'è un'interfaccia? A cosa serve? Nel caso dei tipi di dato astratto, qual è il ruolo dell'interfaccia?
- Si parli del passaggio per valore e per riferimento. In quali casi è preferibile l'uno all'altro? Si parli in concreto del linguaggio C: quali meccanismi di passaggio dei parametri vengono implementati, e come?
- Cosa vuol dire "effetto collaterale" di una funzione? Si facciano alcuni esemi.
- Cosa dice il teorema di Jacopini-Böhm? Cosa comporta?

- Quali sono gli attributi di una variabile? Che cos'è un linguaggio tipato? Qual è l'utilità dei tipi?
- Come vengono rappresentati i numeri reali in virgola mobile?
- Si discutano le varie possibilità di rappresentazione di interi, avendo a disposizione 16 bit.
- Si discutano i seguenti fenomeni: overflow, underflow, perdita di precisione.
- Qual è la differenza tra dichiarazione e definizione? A che cosa servono? Come influiscono sulla compilazione dei programmi?
- Cosa vuol dire valutare un'espressione? Che regole si usano in C?
- Cosa significano coercizione, *casting* e *overloading*?
- Qual è la differenza tra variabili dinamiche e automatiche? A cosa serve averle entrambe?
- Cosa significano *scope* e tempo di vita di una variabile? Si facciano esempi che coprano vari casi, utilizzando le convenzioni del linguaggio C.
- Quali sono i possibili problemi legati ai puntatori?
- Qali sono le varie aree di memoria? Che tipi di informazione contengono?
- Come funziona lo *heap*? Chi lo gestisce? Quali sono i meccanismi che ha il programmatore per utilizzare questa area di memoria?
- Come è strutturato un programma C su più file?
- A cosa serve la dichiarazione di tipi? Si faccia un esempio in C.
- A cosa servono le variabili *static* in C? Si faccia un esempio d'uso.
- Quali sono gli elementi di un record di attivazione? Come vengono utilizzati? Quando vengono inizializzati?
- Quali sono i vari tipi di ricorsione? Come si distinguono?
- Qual è la differenza tra un processo computazionalmente ricorsivo e uno iterativo? È possibile implementare il primo mediante una struttura di controllo di iterazione, e il secondo mediante una funzione ricorsiva?
- A cosa serve il "caso base" in una funzione ricorsiva? Si faccia un esempio (con e senza caso base).
- Si mostrino una versione ricorsiva e una versione iterativa del calcolo del prodotto dei primi n numeri positivi.
- Si discutano i criteri di progetto che possono portare a scegliere di implementare un tipo **Lista** come lista collegata piuttosto che come array.
- Si faccia un esempio di definizione di tipo di dato astratto **stack** (senza implementarlo).
- Si facciano alcuni esempi di algoritmi di complessità $\mathcal{O}(1)$, $\mathcal{O}(n)$, $\mathcal{O}(n \log n)$ e $\mathcal{O}(n^2)$.
- Quali sono i parametri della funzione `main`? Come si usano? Si faccia un esempio.
- Si facciano due esempi di comandi C per il preprocessore, e se ne discuta l'utilità.
- A cosa serve il passaggio di funzioni come parametro? Si prenda a esempio la funzione `qsort`.
- Quali sono i livelli in cui è strutturato un computer? Si descriva in sintesi cosa rappresentano.
- Si spieghi che cosè il percorso dati, e come viene controllato.
- Qual è la differenza tra algoritmo e programma?
- Si discuta il livello della microprogrammazione, e i vantaggi/svantaggi delle architetture RISC rispetto a quelle CISC.
- Quali sono le funzionalità tipiche di un sistema operativo?
- Nel 1946, il calcolatore ENIAC si programmava regolando 6000 interruttori multi-posizione. In questo modo, il programma veniva "cablato" nella macchina. Si dica quali sono state alcune tra le evoluzioni principali dal punto di vista architetturale, già presenti nella successiva macchina di Von Neumann.
- La principale innovazione del PDP-8 fu di avere un unico bus. Si dica qual è il vantaggio rispetto alla macchina di Von Neumann, e si discutano le funzioni dei bus nelle moderne architetture.
- Il System/360 introdusse il concetto di multiprogrammazione. Si dica di cosa si tratta e, facendo riferimento al concetto di context switch, si dia un'idea di come viene realizzato.
- Cosa dice la legge di Moore? Quali sono le implicazioni? È corretto dire che, se la legge di Moore dovesse restare valida nei prossimi anni, allora la complessità degli algoritmi non rappresenterà più un problema?

- Si descriva il ciclo di prelievo-decodifica-esecuzione in un'architettura di Von Neumann, facendo riferimento ai registri implicati.
- Si descriva il funzionamento delle istruzioni memoria-registro. Quali registri sono coinvolti? Quali bus sono coinvolti? In che modo?
- Si spieghi brevemente quali sono i due tipi di parallelismo visti a lezione.
- Si faccia l'esempio di una pipeline doppia a cinque stadi, come quella usata nel Pentium. Quali sono i vantaggi?
- Un'architettura superscalare contiene più unità funzionali di esecuzione all'interno di una stessa pipeline. Perché? Quali sono i vantaggi?
- Array computer, multiprocessori e multicomputer: in cosa si differenziano?
- Si consideri una memoria RAM di 4Kbit, organizzata con parole di 1 byte. Quanti bit di indirizzo servono per accedere a tutte le celle? Se si raddoppiano i bit di indirizzo, lo spazio indirizzabile quanto diventa?
- Si spieghi a cosa serve la memoria cache: quando è preferibile rispetto a soluzioni con memoria su CPU o rispetto alla soluzione con la semplice RAM?
- Si discuta il principio di località, hit-ratio, miss-ratio, e in che modo questi fattori contribuiscono alle prestazioni di una cache.
- Si parli della gerarchia delle memorie, fornendo un esempio per ciascun livello della piramide. Quali sono le caratteristiche principali delle memorie?
- Si consideri un Hard Disk con settori di 4Kbyte, tempo medio di latenza di 15ms e tempo di trasferimento di 400Mbps. In condizioni ideali, qual è il tempo medio che occorre a trasferire un file disposto su 3 settori contigui, attraverso un collegamento USB a 480Mbps? Il collegamento rappresenta un collo di bottiglia?
- Il tempo medio di trasferimento di dati su/da Hard Disk si calcola considerando il seek time, la velocità di rotazione, e il transfer rate. Cosa sono?
- A cosa servono i dischi RAID? Si mettano a confronto la robustezza del supporto, la dimensione dello spazio di storage e la velocità di accesso all'informazione.
- Si discuta il concetto di file, di record logico, e i vantaggi/svantaggi dei file binari rispetto ai file di testo.
- Di solito i file hanno accesso sequenziale. Come bisogna strutturare un file per consentire un accesso diretto?
- Che cos'è un processo? Quali operazioni deve eseguire un sistema operativo per gestire i processi?
- Qual è il ciclo di esecuzione della shell di Unix?
- Si spieghi il concetto di filtro in relazione ai comandi di Unix. Cosa sono ridirezione e piping? Come vengono usati a livello di shell?
- Si spieghi il concetto di file in Unix. Si facciano esempi di file di tipi diversi. Qual è la differenza tra nome assoluto e nome relativo?
- Si discuta il concetto di protezione dell'informazione in Unix. Come vengon gestiti utenti e gruppi? come viene gestito l'accesso ai file?
- Quali sono gli stati in cui si può trovare un processo Unix? In quali di essi è attivo? Cosa significa? Quali sono le possibili transizioni?
- Si spieghi come vengono rappresentati i processi in Unix. Che informazioni contiene il PCB?
- Come vengono modificati i diritti di accesso a un file in Unix?
- Si descriva la struttura di un i-node. Che informazioni contiene? A cosa servono?
- Come viene suddiviso un disco per realizzare un file system in Unix? Si parli del meccanismo usato da Unix per recuperare i blocchi fisici di un file.
- Quali sono le tabelle globali per la gestione del file system in Unix? Che informazioni contengono?

Esercizi di analisi

Esercizio di Analisi #1

Si scrivano l'output e il contenuto di `tempfile` a seguito dell'esecuzione del seguente codice:

```
int main() {
    FILE *fp;
    int i=0,j=0;
    char c[3];
    int v[][2]={ {10, 6 },
                 { 7, 4 },
                 { 4, 2 },
                 { 1, 0 } };

    fp=fopen( "tempfile", "w" );
    while( z[i][0]%z[i][1] )
        fprintf( fp, "%d", z[i++][(j++)%2] );
    fclose( fp );

    fp=fopen( "tempfile", "rb" );
    while( fread( c, sizeof( char ), 2, fp ) )
        printf( "%c", c[0] );
    fclose( fp );
}
```

Esercizio di Analisi #2

Si rappresenti l'evoluzione dello stack e si dica qual è l'output del seguente programma (output a video e risultato restituito dal `main()`).

```
float dummy_f( float x, int n ) {
    switch ( n% 4 ) {
        case 0:
            return n;
        case 1:
            return dummy_f( x*2, n+5 );
        default:
            return dummy_f( x, n+2 );
    }
}

int main () {
    printf("%f", dummy_f( 1.0, 1 ) );
    return 0;
}
```

Esercizio di Analisi #3

Si rappresenti l'evoluzione dello stack e si dica qual è l'output del seguente programma (output a video e risultato restituito dal `main()`).

```
int f( char *s, char Ch ) {
    if( *s=='\0') return 1;
    else if( *s>=Ch ) {
        printf( " X " );
        return f( s+1, Ch+1 );
    }
    else if( strlen( s )>3 ) {
        printf( " %c ", Ch );
        return f( s+3, Ch );
    }
    else return 2*f( s+1, Ch-1 );
}

int main() {
    int R;
    R=f( "marble", 'g' );
    return R;
}
```

Esercizio di Analisi #4

Si rappresenti l'evoluzione dello stack e si dica qual è l'output del seguente programma (output a video e risultato restituito dal `main()`).

```
int R;

int f( int i, int j ) {
    int k=(i+1)%2,
        l=(j+1)%3;
    static int c[2][3] =
        { {0,2,0}, {1,0,4} };
    c[i][j] += c[k][l];
    printf( "%d", c[i][j] );
    if( ( c[i][j]%5 )==0 ) return c[i][j];
    else return f( k, l )+j;
}

int main() {
    R=f( 1,0 );
    return R;
}
```