Fondamenti L-A 2006/2007 per Ing. Elettronica e delle Telecomunicazioni

Terzo scritto, Martedì 16 Gennaio 2007, ore 15, Aule 6.1–6.2

Nome:	**Cognome:**
Matricola:	**Compito:**

(scrivere i propri dati in stampatello)

Modalità di svolgimento della prova

- Scrivere nell'intestazione di questo foglio il proprio nome, cognome e numero di matricola.
- Utilizzare per le risposte i fogli protocollo allegati. Scrivere a penna. Scrivere su ciascun foglio protocollo il proprio nome, cognome e numero di matricola.
- Sui fogli protocollo, per ciascuna risposta indicare in modo chiaro il numero dell'esercizio corrispondente (es. *"esercizio 9"*).
- È assolutamente vietato consultare libri, appunti, manuali, o altro materiale didattico.
- Per tutta la durata dell'esame è necessario tenere spenti tutti i dispositivi elettronici e di comunicazione (cellulari, calcolatrici, palmari, smartphone, laptop, player, ...)
- Nell'ultima mezz'ora di prova non è consentito uscire dall'aula (nemmeno per andare in bagno).
- Per le parti non di codice, si accettano soluzioni scritte in italiano, inglese, portoghese, francese, spagnolo o turco.
- Al termine della prova, *consegnare assieme ai fogli protocollo anche il testo dell'esercizio.*

Tabella degli operatori in C

Precedenza	Operatori	Associatività
1	`() [] -> .`	a sinistra
2	`! ++ -- & *`	a destra
3	`* / %`	a sinistra
4	`+ -`	a sinistra
6	`< <= > >=`	a sinistra
7	`== !=`	a sinistra
11	`&&`	a sinistra
12	`||`	a sinistra
13	`?...:`	a destra
14	`= += -= *=`	a destra
15	`,`	a sinistra

Modalità di valutazione

Il compito si divide in quattro parti (due di teoria, analisi, progetto+implementazione). *Non è necessario rispondere a tutte le domande*, ma per superare l'esame è necessario ottenere una valutazione superiore alla *soglia minima* in ciascuna parte. Il *massimo di punti ottenibili* in ciascuna parte è: 8 per le domande di teoria, 6 per la parte di analisi, 19 per la parte di progetto e implementazione. Il voto complessivo, se sufficiente, sarà compreso tra 18 e 33:

Parte A: Teoria #1 [4 punti]
Soglia minima 2 punti

1. [punti $\in \{-1, +4\}$] **Architettura**
2. [punti $\in \{-1, +4\}$] **Linguaggi**

Parte B: Teoria #2 [4 punti]
Soglia minima 2 punti

3. [punti $\in \{-1, +4\}$] **Funzioni**
4. [punti $\in \{-1, +4\}$] **Run-time**

Parte C: Analisi [6 punti]
Soglia minima 3 punti

5. [punti $\in \{-1, +3\}$] **Output di funzione**
6. [punti $\in \{-1, +6\}$] **Evoluzione dello stack**

Parte D: Progetto & Implementazione [19 punti]
Soglia minima 9 punti

7. [punti $\in \{-1, +7\}$] **Progetto della soluzione**
8. [punti $\in \{-1, +1\}$] **Prototipi e dichiarazioni di tipo**
9. [punti $\in \{-1, +2\}$] **Implementazione** `menu`
10. [punti $\in \{-1, +6\}$] **Implementazione funzione #1**
11. [punti $\in \{-1, +7\}$] **Implementazione funzione #2**

Prototipi delle funzioni di uso comune

Header	Interfaccia	Error
string.h	`size_t strlen( char * ); // size_t e' un tipo int`	
string.h	`char *strcpy( char *, const char * ); // copia`	
string.h	`char *strcat( char *, const char * ); // concatenazione`	
string.h	`int strcmp( const char *, const char * ); // confronto`	
stdlib.h	`void *malloc( size_t ); // restituisce un generico puntatore`	`NULL`
stdlib.h	`void free( void * );`	
stdio.h	`FILE *fopen( char* name, char *mode );`	`NULL`
stdio.h	`int fclose( FILE * );`	`EOF`
stdio.h	`int feof( FILE * ); // vero se file pointer su EOF`	
stdio.h	`int fseek( FILE *, long offs, int orig ); // 0 SEEK_SET, 1 SEEK_CUR, 2 SEEK_END`	
stdio.h	`void rewind( FILE * );`	
stdio.h	`long ftell( FILE * ); // byte a cui si trova il file pointer`	`-1`
stdio.h	`int fprintf( FILE *, char * [, ...] );`	
stdio.h	`int fscanf( FILE *, char * [, ...] );`	
stdio.h	`char *fgets( char *, int, FILE * ); // legge la riga intera fino a '\n'`	`NULL`
stdio.h	`int *fputs( char *, FILE * ); // scrive una stringa e aggiunge '\n'`	`EOF`
stdio.h	`int fread( void *vet, int size, int n, FILE *fp );`	
stdio.h	`int fwrite( void *vet, int size, int n, FILE *fp );`	

Parte A: Teoria #1

Massimo una pagina

1. Si descriva in modo puntuale e conciso l'architettura della Macchina di Von Neumann, con particolare riferimento alla struttura interna della CPU, e si faccia un esempio di lettura o scrittura da/su RAM.
2. Si spieghi come si definisce un linguaggio di programmazione. Si discuta il concetto di linguaggi di programmazione di basso livello/di alto livello, con degli esempi. Si collochino nel giusto contesto linguaggi quali il C, il linguaggio macchina, e l'assembler, e si dica qual è o quali sono le caratteristiche salienti che rendono un linguaggio "di alto livello."

Parte B: Teoria #2

Massimo una pagina

3. Parametri formali e parametri effettivi. Si dica cosa sono, a cosa servono, in che relazione sono tra loro, e si spieghino i concetti di binding e di interfaccia di una funzione.
4. Lo spazio di indirizzamento in C: si discuta in che modo viene partizionata la memoria dedicata all'esecuzione di un programma, e si illustri in quali aree sono allocate (1) le variabili *globali*, (2) le variabili *locali* alle funzioni, (3) le variabili `static`, (4) i *parametri formali* delle funzioni, (5) le variabili *dinamiche*, e (6) il *codice* delle funzioni.

Parte C: Analisi

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

```
FILE *fp; int i=0,j=0; char c[3];
int v[][2]={{6,23},{4,1},{3,22},{2,8}};
fp=fopen( "tempfile", "w" );
while( v[i][1]%v[i][0] ) {
   i++; j++; j=j%2;
   fprintf( fp, "%d", v[i][j] );
}
fclose( fp );
fp=fopen( "tempfile", "rb" );
while( fread( c, sizeof( char ), 2, fp ) )
   printf( "%c", c[0] );
fclose( fp );
```

6. Si descrivano l'evoluzione dello stack e l'output prodotto dall'esecuzione del seguente programma:

```
void r(void) {
   static int count=1;
   printf(" * R%d *\n", count++);
}
int f(char *s, int n) {
   int i=0;
   r();
   if(strlen(s)>n) {
      i=f(s+2,n-1);
      printf("%c ", s[i]);}
   else printf(".\n");
   return strlen(s)-i;}
int main() {
   printf("%d\n",f("effigie",5));
}
```

Parte D: Progetto & Implementazione

Una piccola biblioteca vuole adottare un sistema informatizzato per la gestione dei prestiti di libri. Per far ciò, intende catalogare ciascun libro per autore e titolo, e tener traccia del numero di volumi in possesso, del numero di volumi disponibili (cioè attualmente non concessi in prestito), e del tipo di prestito consentito. Quest'ultima informazione deve distinguere tra:

(a) libri in sola consultazione (non viene concesso prestito);
(b) libri per cui viene concesso un prestito di 2 settimane;
(c) libri per cui viene concesso un prestito di 3 mesi.

La biblioteca vi chiede di sviluppare un programma (database) che possa essere utilizzato per inserire e gestire i dati nel seguente modo:

(a) l'utente deve essere in grado di **inserire un libro nel database**: autore, titolo, ed eventualmente numero di volumi e tipo di prestito. **NOTA:** attraverso autore e titolo, il programma deve essere in grado di riconoscere se un libro esiste già: in tal caso, invece che aggiungere un libro, deve solo incrementare il numero di volumi in possesso;
(b) il programma deve consentire la **gestione dei prestiti** nel seguente modo:
   - deve innanzitutto distinguere tra presa in prestito (P) e restituzione (R);
   - nel caso di prestito (caso "P"), dato autore e titolo deve, cercare il libro in catalogo, se esiste deve segnalare la disponibilità e se il libro è disponibile aggiornare la disponibilità, altrimenti segnalare la non disponibilità. Se il libro non esiste in catalogo, invece, deve segnalare l'assenza del libro dal catalogo;
   - nel caso di restituzione (caso "R"), dato autore e titolo, una volta recuperato il libro dal catalogo deve aggiornarne la disponibilità;
(c) infine, deve consentire la **cancellazione di un libro** dal catalogo: ciò avviene impostando a zero il numero di volumi in possesso.

Resta inteso che, all'uscita, il programma deve essere in grado di conservare in modo permanente le aggiunte/modifiche eventualmente apportate, e che in fase di avvio deve essere in grado di recuperare lo stato del catalogo salvato in precedenza.

7. Si descriva in modo chiaro e succinto il progetto di una possibile soluzione software di tale problema: struttura della soluzione, tipo e formato di file utilizzati, funzioni e relativi input/output, variabili globali se usate, descrizione a parole o mediante diagramma di flusso dei punti salienti degli algoritmi che si intendono utilizzare.
8. Si forniscano i prototipi in C delle funzioni ed eventuali dichiarazioni di tipo necessarie all'implementazione della soluzione proposta.
9. Si implementi una funzione `menu` da usare per scegliere la funzionalità da far eseguire al programma
10. Si implementi una funzione `leggiFile` da utilizzare in una ipotetica soluzione al problema proposto, in cui vengano lette da un file di input e predisposte in opportune strutture dati tutte le informazioni necessarie all'elaborazione dei dati stessi da parte del programma.
11. Si implementi una funzionalità a scelta tra la (a) e la (b).