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

Seconda prova parziale, Martedì 7 Dicembre 2006, ore 15, Aule 6.1–6.2

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

(scrivere i propri dati in stampatello)

Modalità di svolgimento della prova
(invariate rispetto alla prova in itinere)

- 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
(invariate rispetto alla prova in itinere)

Il compito si divide in tre parti (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: 7 per le domande di teoria, 6 per la parte di analisi, 20 per la parte di progetto e implementazione. Il voto complessivo, se sufficiente, sarà compreso tra 15 e 33:

Parte A: Teoria [7 punti]
Soglia minima 4 punti

1. [punti $\in \{-1, +2\}$] **Record & puntatori**
2. [punti $\in \{-1, +2\}$] **Funzioni**
3. [punti $\in \{-1, +2\}$] **Run-time**
4. [punti $\in \{-1, +2\}$] **Binding**
5. [punti $\in \{-1, +1\}$] **File**

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

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

Parte C: Progetto & Implementazione [20 punti]
Soglia minima 9 punti

8. [punti $\in \{-1, +6\}$] **Progetto della soluzione**
9. [punti $\in \{-1, +1\}$] **Dichiarazioni**
10. [punti $\in \{-1, +2\}$] **Implementazione main**
11. [punti $\in \{-1, +5\}$] **Implementazione funzione #1**
12. [punti $\in \{-1, +8\}$] **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. Discutere brevemente la gestione delle variabili dinamiche in C. Che funzioni usare? Come? Che visibilità hanno le variabili dinamiche? Che tempo di vita? Fare un esempio con una variabile dinamica di tipo vettore di 10 float.
2. Di cosa consta l'interfaccia di una funzione? A cosa serve? Come viene effettuata in C la dichiarazione di una funzione? Fare un esempio.
3. Che cos'è lo spazio di indirizzamento? Si spieghi come viene allocata la memoria ai processi nel modello del C, e nello specifico quali parti sono allocate all'avvio del programma e quali invece hanno un'allocazione dinamica.
4. Cosa succede all'atto della chiamata di funzione? Si discuta dell'aspetto del binding, tramite un esempio di funzione che contiene due parametri di tipo puntatore a float.
5. Come avviene l'accesso ai file nel linguaggio C? Che cos'è un "record logico" di un file? si mettano a confronto il caso dei file binari e quello dei file di testo.

Parte B: Analisi

6. Si scrivano l'output e il contenuto di `tempfile` a seguito dell'esecuzione del seguente codice:
```
FILE *fp; int i=0; char c[3];
int v[]={6,5,5,4,3,2,1};
fp=fopen( "tempfile", "w" );
while( v[i]-2 )
   fprintf( fp, "%d", v[i++] );
fclose( fp );
fp=fopen( "tempfile", "rb" );
while( fread( c, sizeof( char ), 2, fp ) )
   printf( "%c", c[0] );
fclose( fp );
```
7. Si descrivano l'evoluzione dello stack e l'output prodotto dall'esecuzione del seguente programma:
```
int f( int x, char c ) {
   if( x<5 ) return 1;
   if( x<10 ) return x;
   printf( "%c ", c+(x%5) );
   return f( x/2, c )+f( x-7, c );
}
int main() {
   printf( "%d", f( 23, 'l' ) );
}
```

Parte C: Progetto & Implementazione

La *Radioactive Brothers Inc.* (un negozio di vendite online di isotopi radioattivi) vi ha contattato per effettuare alcune decisioni strategiche circa il rinnovamento del catalogo dei propri prodotti. Vi ha fornito una pagina html, da prendere come modello di esempio di file contenente i dati sui prodotti attualmente disponibili in stock. In questo file, le prime 3 righe e l'ultima contengono del codice html (che non vi interessa), mentre le righe intermedie contengono ciascuna per ciascun isotopo in stock il nome dell'elemento, l'isotopo (in intero), la quantità in stock, l'attività (un numero reale), la provenienza, e il prezzo (in dollari). Un esempio di file html (un file di testo) è il seguente:
```
<html xmlns="http://www.w3.org/1999/xhtml">
<head> <title>RBI Isotopes</title> </head>
<body bgcolor="white">
   Polonium 210 27 0.1 Rome 69.00<br />
   Cesium 137 2 10.0 Baku 100.00<br />
   Strontium 90 10 0.1 Dallas 69.00<br />
</body> </html>
```
Si noti che ciascuna riga (a parte quelle che non interessano) è terminata dalla stringa `<br />`. La seconda riga, ad esempio, indica che sono in stock due unità di Cesio^{137}, con un'attività di $10.0\mu Ci$, provenienti da Baku, al prezzo unitario di 100$. La *RBI* vi chiede di produrre un programma che sia in grado di leggere file html in tale formato, ed estrarre informazioni di vario genere. Il nome del file html da cui estrarre informazioni deve essere fornito in input al programma. Le informazioni che si vogliono ottenere sono le seguenti:

(a) elenco (senza ripetizioni) degli isotopi in stock;
(b) se esistono, quali sono gli isotopi i cui campioni hanno tra di loro una differenza significativa di attività (dal più debole al più forte differenza di almeno un ordine di grandezza)
(c) i nomi dei due isotopi che hanno la provenienza più differenziata (cioè, ciascun isotopo, tipo il Polonio^{210} o il Cesio^{137}, in generale sarà disponibile con varie provenienze: si vogliono elencare i due isotopi che tra tutti hanno il maggior numero di provenienze distinte)

8. Si descriva in modo chiaro e succinto il progetto di una possibile soluzione software di tale problema: struttura della soluzione, 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.
9. Si forniscano le dichiarazioni in C di tutte le funzioni ed eventuali strutture dati globali necessarie all'implementazione della soluzione proposta.
10. Si implementi una funzione `menu` da usare per scegliere la funzionalità da far eseguire al programma
11. Si implementi una funzione da utilizzare in una possibile soluzione al problema proposto, in cui vengano lette dal file di input e predisposte nelle opportune strutture dati tutte le informazioni necessarie all'elaborazione dei dati stessi da parte del programma.
12. Si implementi in modo completo una funzione che risponda al punto (*b*) o al punto (*c*). *(Se la funzione si basa su dati organizzati su strutture dati interne al programma in un certo modo, è necessario anche fornire l'implementazione delle funzioni/procedure che effettuano tale organizzazione — a meno di quanto già mostrato al punto precedente.)*