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

Secondo scritto, Martedì 19 Dicembre 2006, ore 15, Aula 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 15 e 33:

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

1. [punti $\in \{-1, +3\}$] **Architettura**
2. [punti $\in \{-1, +3\}$] **Linguaggi**

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

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

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

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

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

8. [punti $\in \{-1, +6\}$] **Progetto della soluzione**
9. [punti $\in \{-1, +1\}$] **Prototipi e dichiarazioni di tipo**
10. [punti $\in \{-1, +2\}$] **Implementazione** `menu`
11. [punti $\in \{-1, +5\}$] **Implementazione funzione #1**
12. [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

1. Si descriva il ciclo di fetch / decode / execute della CPU: cosa avviene in ciascuna fase, e quali componenti dell'architettura di Von Neumann sono coinvolti.
2. La traduzione di un programma: si dica quali sono le differenze tra intepreti e compilatori.

Parte B: Teoria #2

3. Che cosa sono i riferimenti pendenti (*dangling references*)? Possono costituire un problema? Come? Cosa si può fare per evitarlo?
4. Variabili globali e variabili static. Si discutano tempo di vita, visibilità, e loro allocazione in memoria nel modello seguito dal C.
5. Si consideri il problema della scrittura di 10 record su file binario (nello specifico, si supponga che i record costituiscono gli elementi di un vettore in memoria, e devono essere scritti su file binario). Spiegare con un esempio quali sono le strutture dati necessarie per effettuare tale operazione di scrittura all'interno di un programma in C.

Parte C: Analisi

6. 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]={{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 );
```
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 2;
   if( x<10 ) return x;
   printf( "%c ", c+(x%5) );
   return f( x/2, c )+f( x-8, c );
}
int main() {
   printf( "%d", f( 24, 'p' ) );
}
```

Parte D: Progetto & Implementazione

La Società Amazzonica per lo Studio e la Conservazione degli Alligatori ($SASCA$) ha raccolto, negli ultimi 5 mesi, una serie di informazioni circa le abitudini di una ventina di esemplari campione. Tali informazioni sono state salvate in un file binario, `alli.gator`, strutturato a record così composti:

- un codice alfanumerico di 5 caratteri, di cui il primo identifica univocamente l'esemplare a cui si riferiscono le informazioni, gli altri quattro invece rappresentano il codice di una determinata situazione che è stata osservata sul campione;
- un intero $\in \{1, \dots 10\}$ che indica il codice della zona geografica in cui si trovava l'alligatore quando è stata raccolta l'osservazione;
- un campo numerico che rappresenta il giorno in cui è stata effettuata l'osservazione (a partire da un certo giorno 0).

La $SASCA$ vi ha commissionato un programma, che intende usare al fine di estrapolare informazioni utili a capire il comportamento degli alligatori, a partire dal campione di dati raccolti. Dato un certo giorno in input (sotto forma di numero intero, e.g. 140 indica il 140^o giorno a partire dal giorno 0), il programma deve essere in grado di fornire risposte alle seguenti interrogazioni:

(a) in quante diverse zone geografiche sono state osservate situazioni relative ad alligatori negli ultimi 60 giorni;
(b) in quali giorni si è verificato il massimo numero di osservazioni (indicare un giorno solo se si ha un unico giorno con un tale numero di osservazioni, altrimenti indicare tutti i giorni che soddisfano *ex-aequo* tale requisito)
(c) codice dell'esemplare su cui è stata effettuata una certa osservazione un numero di volte maggiore che non su qualsiasi altro esemplare. (Il codice dell'osservazione è dato in input come stringa di 4 caratteri)

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 i prototipi in C di tutte le funzioni ed eventuali dichiarazioni di tipo 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.)*