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

Sesto scritto, Martedì 11 Settembre 2007, ore 9, 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"*).
- assolutamente vietato consultare libri, appunti, manuali, o altro materiale didattico.
- Per tutta la durata dell'esame necessario tenere spenti tutti i dispositivi elettronici e di comunicazione (cellulari, calcolatrici, palmari, smartphone, laptop, player, ...)
- Nell'ultima mezz'ora di prova non 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, una di analisi, una di progetto e implementazione). *Non è necessario rispondere a tutte le domande*, ma per superare l'esame è necessario ottenere una valutazione sufficiente (cioè superiore alla *soglia minima*) in ciascuna parte.

Il *massimo di punti ottenibili* in ciascuna parte è: 4 per ciascuna sezione 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 [4 punti]
Soglia minima 2 punti

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

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

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

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 $\in \{-1, +13\}$] **Implementazione delle funzioni**

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

Massimo una pagina

1. Si spieghi cosa si intende per traduzione e interpretazione di programmi, e si discutano le differenti implicazioni dei due approcci dal punto di vista teorico e pratico.
2. Si scriva una grammatica BNF da utilizzare per la produzione di stringhe per la dichiarazione di variabili di tipo intero, puntatore a intero, vettore e matrice di interi: `int a`, `int vet1[43]`, `int w2[43][2][5]`, `int *p`, etc. Si assuma che gli identificatori contengano solo cifre [0-9] e caratteri [a-z] e che comincino con un carattere, che le dimensioni del vettore siano sempre specificate (come nell'esempio: `[43]`, `[43][2][5]`, ..., non `[]`, `[][2][5]`, ...) e che i vettori possano essere N-dimensionali con N grande a piacere (nell'esempio, vet1 è 1-dimensionale, w2 è 3-dimensionale).

Parte B: Teoria

Massimo una pagina

3. Si spieghi il concetto di puntatore, e il suo utilizzo per la gestione di variabili dinamiche tramite `malloc` e `free`. Si discuta un esempio in cui viene definita una variabile dinamica all'interno di una funzione, e viene modificato il valore di tale variabile all'interno di un'altra funzione.
4. Si discutano i concetti di funzione, codice della funzione, istanza della funzione. Si spieghino gli elementi essenziali di una funzione (identificatore, parametri, corpo, valore in uscita), e si discuta il comportamento di un processo a *run-time* all'atto della chiama a funzione.

Parte C: Analisi

5. Si scriva l'output prodotto dall'esecuzione del seguente programma:

```
int v[][3] = {{1,2,3},{4,5,6},{7,8,9}};

void f( int *a, int *b ) {
   int i=*a, j=*b;
   v[i][2-j] = v[2-i][j]+10;
   *a = *a + 1;
   *b = *b + 2;
}

int main() {
   int i=0,j=0;
   while( i<3 && j<3 ) {
      f( &i, &j );
   }
   for(i=0;i<3;i++)
      for(j=0;j<3;j++)
         printf( "%d\n", v[i][j] );
}
```

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

```
int v[][3] = {{1,2,3},{4,5,6},{7,8,9}};

void f( int *a, int *b ) {
   int i=*a, j=*b;
   if(i+j>1) return 10;
   *a = *a + 1;
   v[i][2-j] = v[2-i][j]+f(b,a);
   return v[i][j];
}

int main() {
   int i=0,j=0;
   f( &i, &j );
   for(i=0;i<3;i++)
      for(j=0;j<3;j++)
         printf( "%d\n", v[i][j] );
}
```

Parte D: Progetto & Implementazione

Si vuole creare un programma che fornisca delle statistiche relative al funzionamento di un veicolo. I dati di input per le statistiche sono prodotti dal veicolo stesso, che trasmette, a ogni kilometro percorso:

- il kilometraggio (0-200.000 Km),
- il momento in cui il dato è prodotto (sotto forma di un numero intero che rappresenta il numero di secondi trascorsi dal momento in cui il veicolo è stato prodotto),
- il livello di carburante (0-60 litri).

I dati trasmessi sono ricevuti da un computer che li salva sotto forma di record, in modo incrementale, aggiungendoli in coda a un file binario `vehicle.dat`.
Il programma deve fare uso dei dati contenuti nel file `vehicle.dat` aggiornato per produrre, alla richiesta dell'utente, una delle seguenti informazioni:

(a) numero totale di kilometri percorsi;
(b) stima del numero totale di litri di carburante consumati dal veicolo;[1]
(c) stima del numero di kilometri che il veicolo può percorrere con il carburante a disposizione;
(d) velocità media calcolata negli ultimi 200 kilometri.

7. Si imposti la soluzione. Si discutano le eventuali scelte di progetto. Si fornisca una descrizione accurata delle funzioni, degli algoritmi e delle strutture dati necessari per implementare il programma.
8. Si forniscano i prototipi in C delle funzioni ed eventuali dichiarazioni di tipo necessarie all'implementazione della soluzione proposta.
9. Si implementi in C una funzione `menu` da usare per selezionare la funzionalità da far eseguire al programma e per terminare l'esecuzione del programma.
10. Si scriva il codice in C di una o più funzioni che implementino i punti a, b, c, e d.

[1] Sarà necessario tener conto degli eventuali rifornimenti di carburante. Suggerimento: il livello di carburante indicato nel record relativo al kilometro successivo a ciascun rifornimento è maggiore del livello di carburante indicato nel record precedente.