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

Quarto scritto, Martedì 27 Marzo 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"*).
- 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, 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 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 #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 $\in \{-1, +4\}$] **Visibilità**
4. [punti $\in \{-1, +4\}$] **Passaggio per riferimento**

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 l'architettura di Von Neumann, e in particolare si spieghi il ciclo della CPU per l'esecuzione delle istruzioni.
2. Si scriva una grammatica BNF da utilizzare per la produzione di stringhe nel formato: `N123.`, `M132.`, `M1234.`, `N1114.`, `N3333.`, ...(`N` o `M` seguita da tre o quattro cifre nell'intervallo [1..4], seguite da `.`).

Parte B: Teoria #2

Massimo una pagina

3. Si discuta il concetto di visibilità degli identificatori in C: si considerino in particolare identificatori di variabile e di funzione, e si enuncino le regole che definiscono la loro visibilità.
4. Passaggio per riferimento: a cosa serve, come viene realizzato in C, se ci sono dei casi in cui è "automatico." Si faccia un esempio, in cui sia chiaro qual è il prototipo del servitore, e come viene eseguita l'invocazione del servitore da parte del cliente.

Parte C: Analisi

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

```
int f( int j, char c ) {
   return j - printf( "%c", c
);
}

int main() {
   int i=4;
   while( i>0 ) {
      printf( "%d\n", i=f( i, 'C' +i )
);
   }
}
```

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

```
int start[]={3,4,5};
int fun( int *pin, int i )
{
   int *pout=NULL, j=i,
sum=pin[0];
   if( i>0 ) {
      pout=( int * )malloc( i*sizeof( int )
);
      while( j>0 ) {
         pout[j-1] = pin[j];
         sum += pin[j--];
}
      printf( "%d\n", fun( pout, i-1)
);
      free( pout );
   }
   return sum;
}

int main() {
   fun( start, 2 );
}
```

Parte D: Progetto & Implementazione

Un'agenzia di intelligence vi ha commissionato un prototipo di sistema informativo allo scopo di scongiurare possibili atti terroristici. L'idea dell'agenzia è quella di creare un database di tutti gli individui in transito negli aeroporti, e in base alle informazioni nel database, assegnare un valore di pericolosità $\mathcal{P}$ a ciascun individuo. La pericolosità (da $\mathcal{P} = 0$, minima, a $\mathcal{P} = 10$, massima) viene stimata in base al seguente algoritmo:

(a) se la coppia (cognome,nome) dell'individuo coincide con una coppia (cognome,nome) trovata all'interno di un file `blacklist.txt`, $\mathcal{P} = 10$;
(b) altrimenti, se (cognome,nome) dell'individuo coincidono con una coppia (cognome,nome) trovata all'interno di un file `whitelist.txt`, $\mathcal{P} = 0$;
(c) altrimenti $\mathcal{P} = 5 + x_1 + x_2 + x_3$, dove:

- $x_1 = 1$ se stato civile: single, $x_1 = -1$ se stato civile = coniugato, $x_1 = 0$ in tutti gli altri casi;
- $x_2 = 1$ se occhi scuri, -1 se occhi chiari, 0 in tutti gli altri casi;
- $x_3 = 1$ se fronte bassa, -1 se fronte alta, 0 in tutti gli altri casi;

L'agenzia vi chiede di sviluppare un programma per gestire i dati necessari. Nella fattispecie, i due file `blacklist.txt` e `whitelist.txt` sono dati, e hanno un formato testuale; mentre il programma deve utilizzare un file `individui.dat` (che può essere, a scelta, testuale o binario) che contenga per ciascun individuo cognome, nome, stato civile, colore degli occhi, e altezza della fronte (alta/bassa/sconosciuta/...). Le funzionalità di base richieste sono:

(a) **inserimento** di un nuovo individuo (si faccia l'ipotesi che non vi siano omonimi, vale a dire che se viene trovato un individuo con lo stesso cognome e nome, il programma deve eseguire un semplice **aggiornamento** dei dati esistenti sull'individuo con i nuovi valori);
(b) **visualizzazione** delle informazioni riguardanti un individuo dati cognome e nome (senza omonimie) e della sua relativa pericolosità $\mathcal{P}$ calcolata in base all'algoritmo di cui sopra;
(c) **cancellazione** di un individuo da `individui.txt`.

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 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).