Esame di Fondamenti di Informatica
Ingegneria Meccanica (A–O)
Appello del 19/6/2001

Compito A

Esercizio 1

Un elaboratore adotta per i **numeri interi** una rappresentazione in complemento a due su un byte e per i **numeri reali** una rappresentazione in virgola mobile con un byte per la mantissa normalizzata in segno/modulo (si usi il primo bit della mantissa per rappresentare il segno) ed un byte per l'esponente in complemento a due.
Si consideri l'espressione:

2.82 – (37 – 26)

Indicare il risultato dell'espressione ottenuto eseguendo il calcolo con l'elaboratore dato. Mostrare i passaggi intermedi eseguiti dall'elaboratore con particolare riferimento alle operazioni ed alla rappresentazione interna in binario, nonché i relativi eventuali errori compiuti. Scrivere il risultato finale in codifica decimale.

Esercizio 2

Dato il seguente programma C:

```
#include <stdio.h>

int V[] = {1,2,3,4};
int A[] = {0,0,0,0};
int B[] = {1,1,1,1};

int F(int V[], int n) {
  int i;

  for(i=0; i<n; i++) V[i]=i%n;
  V=B;
  for(i=n; i>0; i--) V[i]*=i;
  return V[i];
}

main() {
  int i=3;

  printf("%d\n", F(A, i));
  for(i=0; i<4; i++) printf("%d\t%d\n", A[i], B[i]);
}
```

Scrivere l'output prodotto dal programma ed illustrare la dinamica dei record di attivazione sullo stack. Motivare opportunamente le risposte.

Esercizio 3

Per la gestione dei propri ombrelloni il bagno "Marco" utilizza un file binario chiamato "BAGNO.DAT", all'interno del quale sono memorizzati, per ogni ombrellone, il cognome della persona che ha eventualmente prenotato l'ombrellone per tutta la stagione, il numero di sdraio desiderate e la richiesta o meno di una cabina (variabile booleana che vale 0 se e solo se il cliente non ha richiesto l'uso della cabina). Un ombrellone è considerato libero se e solo se il nome del cliente è una stringa vuota. Si supponga che il numero massimo di ombrelloni sia 100.
Le richieste di prenotazione vengono, invece, memorizzate all'interno di un file di testo chiamato "DOMANDE.TXT", nel quale ogni riga contiene il cognome del cliente, il numero di sdraio desiderate ed un valore intero che indica la richiesta o meno della cabina. Si supponga che ogni cliente prenoti un solo ombrellone per la stagione e che non esistano omonimie tra i clienti. Si supponga inoltre che le richieste non superino mai il numero di ombrelloni disponibili.
Si scriva un programma C per la gestione degli ombrelloni del bagno "Marco" che:

1. Legga dal file "BAGNO.DAT" le informazioni sugli ombrelloni e le memorizzi all'interno di un vettore `V`.
2. Per ogni riga del file "DOMANDE.TXT", aggiorni il vettore `V` inserendo i dati del cliente richiedente nel primo ombrellone libero presente nel vettore. A tal scopo si supponga l'esistenza di una funzione `primo_libero` che, dato un vettore di ombrelloni, restituisca l'indice corrispondente al primo ombrellone libero del vettore.
3. Letto da tastiera il cognome di un cliente, annulli la prenotazione relativa a tale cliente. A tal scopo si supponga l'esistenza di una procedura `annulla` che, dati un vettore di ombrelloni, il numero di elementi effettivamente contenuti nel vettore ed una stringa che rappresenta il cognome del cliente, aggiorni il vettore in maniera opportuna.
4. Si scriva il codice della funzione `primo_libero`.
5. Si scriva il codice della funzione `annulla` (a tal scopo si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche).

Esercizio 4

Illustrare il concetto di algoritmo.

Soluzione Esercizio 1

Rappresentazione **2.82**:
$(2.82)_{10} = (10.110100\ldots)_2$
Errore di troncamento!
Normalizzando: 0.10110100×2^2
byte esponente = **0000 0010**
byte mantissa = **0011 0100**

Rappresentazione **37**:
$(37)_{10} = (100101)_2$
rappr. interna = **0010 0101**

Rappresentazione **–26**:
$(26)_{10} = (11010)_2$
rappr. interna = 0001 1010
complementando:
rappr. interna = **1110 0110**

Risultato **37 – 26**:
0010 0101 +
1110 0110 =
$(1)0000\ 1011 = (11)_{10}$

Risultato **2.82 – 11**:
Normalizzazione del risultato: 0.1011×2^4
Incolonnamento del numero più piccolo: 0.00101101×2^4
Nessun errore di incolonnamento!
$0.1011\ 0000 \times 2^4$ –
$0.0010\ 1101 \times 2^4$ =
$0.1000\ 0011 \times 2^4$
byte esponente = **0000 0100**
byte mantissa = **1000 0011**
risultato: $-(1000.0011)_2 = -(8.1875)_{10}$
risultato atteso: $-(8.18)_{10}$

Soluzione Esercizio 2

Il risultato del programma è il seguente:

1
0 1
1 1
2 2
0 3

Infatti, la prima parte della funzione F modifica i primi 3 (n=3) valori del vettore A, mentre la seconda parte modifica gli ultimi 3 (n=3) valori del vettore B.

Soluzione Esercizio 3

```
#include <stdio.h>
#include <string.h>

typedef struct {
  char cognome[30];
  int sdraio;
  int cabina;
} ombrellone;

int primo_libero(ombrellone V[]);
void annulla(ombrellone V[], int N, char s[]);

main() {
  int N=0;
  char nome[30];
  ombrellone V[100];
  FILE *fp=fopen("BAGNO.DAT", "rb");
  while(!feof(fp)) {
    fread(&(V[N]), sizeof(ombrellone), 1, fp);
    N++;
  }
/* in alternativa:
  N=fread(V, sizeof(ombrellone), 100, fp); */
  fclose(fp); /* fine domanda 1. */
  fp=fopen("DOMANDE.TXT", "r");
  while(!feof(fp)) {
    int indice=primo_libero(V);
    fscanf(fp, "%s %d %d", V[indice].cognome, &(V[indice].sdraio),
      &(V[indice].cabina));
  }
  fclose(fp); /* fine domanda 2. */
  scanf("%s", nome);
  annulla(V, N, nome); /* fine domanda 3. */
}

int primo_libero(ombrellone V[]) {
  int i=0;
  while(V[i].cognome[0]=='\0') i++;
  return i;
} /* fine domanda 4. */

void annulla(ombrellone V[], int N, char s[]) {
  int i;
  for(i=0; i<N; i++)
    if(!strcmp(V[i].cognome, s)) {
      V[i].cognome[0]='\0';
      break;
    }
} /* fine domanda 5. */
```

Esame di Fondamenti di Informatica
Ingegneria Meccanica (A–O)
Appello del 19/6/2001

Compito B

Esercizio 1

Un elaboratore adotta per i **numeri interi** una rappresentazione in complemento a due su un byte e per i **numeri reali** una rappresentazione in virgola mobile con un byte per la mantissa normalizzata in segno/modulo (si usi il primo bit della mantissa per rappresentare il segno) ed un byte per l'esponente in complemento a due.
Si consideri l'espressione:

5.71 – (35 – 22)

Indicare il risultato dell'espressione ottenuto eseguendo il calcolo con l'elaboratore dato. Mostrare i passaggi intermedi eseguiti dall'elaboratore con particolare riferimento alle operazioni ed alla rappresentazione interna in binario, nonché i relativi eventuali errori compiuti. Scrivere il risultato finale in codifica decimale.

Esercizio 2

Dato il seguente programma C:

```
#include <stdio.h>

int V[] = {4,3,2,1};
int A[] = {0,0,0,0};
int B[] = {1,1,1,1};

int F(int A[], int n) {
  int i;

  for(i=0; i<n; i++) A[i]+=i;
  A=B;
  for(i=n; i>0; i--) A[i]*=V[i];
  return i;
}

main() {
  int i=3;

  printf("%d\n", F(A, i));
  for(i=0; i<4; i++) printf("%d\t%d\n", A[i], B[i]);
}
```

Scrivere l'output prodotto dal programma ed illustrare la dinamica dei record di attivazione sullo stack. Motivare opportunamente le risposte.

Esercizio 3

Per la gestione dei propri ombrelloni il bagno "Marco" utilizza un file binario chiamato "BAGNO.DAT", all'interno del quale sono memorizzati, per ogni ombrellone, una variabile booleana che indica se l'ombrellone è prenotato o meno per tutta la stagione, il cognome della persona che lo ha eventualmente prenotato ed il numero di sdraio richieste. Si supponga che il numero massimo di ombrelloni sia 200. Ogni cliente può prenotare più di un ombrellone.
Le richieste di prenotazione vengono, invece, memorizzate all'interno di un file di testo chiamato "DOMANDE.TXT", nel quale ogni riga contiene il cognome del cliente ed il numero di sdraio richieste. Si supponga che non esistano omonimie tra i clienti e che le richieste non superino mai il numero di ombrelloni disponibili.
Si scriva un programma C per la gestione degli ombrelloni del bagno "Marco" che:

1. Legga dal file "BAGNO.DAT" le informazioni sugli ombrelloni e le memorizzi all'interno di un vettore `V`.
2. Per ogni riga del file "DOMANDE.TXT", aggiorni il vettore `V` inserendo i dati del cliente richiedente nel primo ombrellone libero presente nel vettore. A tal scopo si supponga l'esistenza di una procedura `aggiorna` che, dati un vettore di ombrelloni, il cognome del cliente ed il numero di sdraio richieste, aggiorni il vettore in maniera opportuna.
3. Letto da tastiera il cognome di un cliente, annulli tutte le prenotazioni relative a tale cliente. A tal scopo si supponga l'esistenza di una procedura `annulla` che, dato un vettore di ombrelloni, il numero di elementi effettivamente contenuti nel vettore ed una stringa che rappresenta il cognome del cliente, aggiorni il vettore in maniera opportuna.
4. Si scriva il codice della funzione `aggiorna` (a tal scopo si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima).
5. Si scriva il codice della funzione `annulla` (a tal scopo si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche).

Esercizio 4

Illustrare il concetto di sistema operativo.

Soluzione Esercizio 1

Rappresentazione 5**.71**:
$(5.71)_{10} = (101.10110\ldots)_2$
Errore di troncamento!
Normalizzando: 0.10110110×2^3
byte esponente = **0000 0011**
byte mantissa = **0011 0110**

Rappresentazione **35**:
$(35)_{10} = (100011)_2$
rappr. interna = **0010 0011**

Rappresentazione **–22**:
$(22)_{10} = (10110)_2$
rappr. interna = 0001 0110
complementando:
rappr. interna = **1110 1010**

Risultato **35 – 22**:
0010 0011 +
1110 1010 =
$(1)0000\ 1101 = (13)_{10}$

Risultato **5.71 – 13**:
Normalizzazione del risultato: 0.1101×2^4
Incolonnamento del numero più piccolo: 0.01011011×2^4
$0.1101\ 0000 \times 2^4$ –
$0.0101\ 1011 \times 2^4$ =
$0.0111\ 0101 \times 2^4$
Normalizzando: 0.11101010×2^3
Errore di cancellazione!
byte esponente = **0000 0011**
byte mantissa = **1110 1010**
risultato: $-(111.01010)_2 = -(7.3125)_{10}$
risultato atteso: $-(7.29)_{10}$

Soluzione Esercizio 2

Il risultato del programma è il seguente:

0
0 1
1 3
2 2
0 1

Infatti, la prima parte della funzione `F` modifica i primi 3 (`n`=3) valori del vettore `A`, mentre la seconda parte modifica gli ultimi 3 (`n`=3) valori del vettore `B`.

Soluzione Esercizio 3

```
#include <stdio.h>
#include <string.h>

typedef struct {
  int occupato;
  char cognome[30];
  int sdraio;
} ombrellone;

void aggiorna(ombrellone V[],char s[], int sdraio);
void annulla(ombrellone V[], int N, char s[]);

main() {
  int N=0;
  char nome[30];
  ombrellone V[200];
  FILE *fp=fopen("BAGNO.DAT", "rb");
  while(!feof(fp)) {
    fread(&(V[N]), sizeof(ombrellone), 1, fp);
    N++;
  }
/* in alternativa:
  N=fread(V, sizeof(ombrellone), 200, fp); */
  fclose(fp); /* fine domanda 1. */
  fp=fopen("DOMANDE.TXT", "r");
  while(!feof(fp)) {
    int sdraio;
    fscanf(fp, "%s %d", nome, &sdraio);
    aggiorna(V, nome, sdraio);
  }
  fclose(fp); /* fine domanda 2. */
  scanf("%s", nome);
  annulla(V, N, nome); /* fine domanda 3. */
}

void aggiorna(ombrellone V[],char s[], int sdraio) {
  int i=0;
  while(V[i].occupato) i++;
  V[i].occupato=1;
  strcpy(V[i].cognome, s);
  V[i].sdraio=sdraio;
} /* fine domanda 4. */

void annulla(ombrellone V[], int N, char s[]) {
  int i;
  for(i=0; i<N; i++)
    if((V[i].occupato)&&(!strcmp(V[i].cognome, s)))
      V[i].occupato=0;
} /* fine domanda 5. */
```