Politecnico di Milano - Dipartimento di Elettronica e informazione

Prof. Mauro Negri

Fondamenti di Informatica
Informatica I

III appello 6 settembre 2011

Matricola _____ Cognome_____ Nome_____

Durata prova: 1 ora ½

Istruzioni
Non separare questi fogli. Si può utilizzare la matita.
L'uso di cellulari, libri, eserciziari, appunti o calcolatrici durante lo svolgimento della prova comporta l'annullamento della prova.

Per entrambi i corsi

Esercizio 1 (FI: 6 punti Info1: 9 punti)

Esercizio 2 (FI: 11 punti Info1: 13 punti)

Esercizio 3 (FI: 8 punti Info1: 8 punti)

Solo per fondamenti di Informatica

Esercizio 4 (5 punti)

Punteggio totale (30 punti) _____

Esercizio 1

1. Indicare la rappresentazione in complemento a 2 su 8 bit dei numeri:
   A = + 88 (base 10) 01011000
   B = - 22 (base 10) 11101010

2. Indicare la rappresentazione in base 8 e 16 di dei numeri
   A= +23 (base 10) base 8 = 27 base 16 = 17
   B= -22 (base 10) base 8 = -26 base 16 = -16

3. Indicare quale numero decimale rappresenta la seguente sequenza di bit codificata in complemento a 2:
   1111.1100.0101 -59

4. Data la definizione di una variabile float v= 0,55, descrivere la sua rappresentazione in virgola mobile in base allo standard IEEE754 semplice precisione FP32;
   a) Definire prima il numero nel formato normalizzato intermedio (TRONCARE parte decimale alla terza cifra decimale):

$$1{,}1 * 2^{-1}$$

b) Codificare poi il numero normalizzato del punto a) usando il formato FP32:
S (1 bit di segno): 0
M (23 bit per la mantissa – fermarsi a 6 bit): 000110
E (8 bit per l'esponente): 01111110

5. Con riferimento al numero float codificato all'esercizio precedente indicare, dopo gli arrotondamenti effettuati nei punti a) e b), quale sia il vero valore (in base 10) descritto dalla sequenza di bit memorizzata in S, M ed E del punto b) (si consiglia di esprimere la mantissa in termini frazionari).
35/64

6. Date le seguenti definizione globali int v=1; int p[10];
e le seguenti funzioni:

```
int R(int *a, int b) {
        int local ;
        …..
              (istruzione 1)
        return local;
    }
int main() {v= R(&p, 10);  }
```

Disegnare lo stato dello stack durante l'esecuzione dell'istruzione 1 del programma precedente, indicando cosa conterranno le singole parole di memoria allocate per il RDA di R.

	<- SP
local	<- R0
R0	
Indirizzo per RTS	
10	
&p	
Valore ritornato	

Esercizio 2 Si scriva un programma C per la gestione dei prodotti venduti da una ditta. I dati memorizzati per ogni prodotto sono volume, costo e quantità. I dati dei prodotti sono memorizzati in una lista di elementi dinamici con la seguente struttura.

struct rec {int volume; int costo; int quantita; struct rec *puntV; struct rec *puntC;};
struct rec *listaV, *listaC;
dove:

- il campo puntatore **puntV** punta all'elemento della lista contenente il prodotto immediatamente precedente in ordine di volume e il campo **puntC** punta all'elemento contenente il prodotto immediatamente precedente in ordine di costo (entrambi gli ordinamenti sono quindi per valori decrescenti);
- ListaV e listaC puntano rispettivamente al prodotto di volume più grande e di costo maggiore.

Si scriva un sottoprogramma di nome **Read** che riceve come parametro la lista dinamica dei prodotti (il sottoprogramma suppone che la lista sia vuota e quindi non la controlla) e provvede ad aprire il file "a.dat" (nome ricevuto come parametro) contenente una sequenza di record di tipo **struct rec1 {int volume; int costo; int quantita;};** i record sono ordinati all'interno del file per valori di volume decrescente. Il sottoprogramma procede a leggere ogni record del file e a caricarlo in una variabile dinamica e infine a inserire l'elemento creato nella lista dinamica utilizzando i valori di volume (quindi inizializzando il valore di puntV). Alla fine restituisce la lista caricata tramite un parametro. Il caricamento non considera quindi i valori di costo e pertanto il puntatore puntC di ogni elemento e la variabile ListaC saranno inizializzati tutti a NULL, mentre utilizzerà il puntatore puntV per collegare gli elementi in ordine di volume. Il sottoprogramma controlla l'apertura corretta del file e l'allocazione corretta della memoria e se non esistono errori ritorna 1 altrimenti ritorna 0, restituendo in questo caso la lista nello stato in cui si trovava all'atto dell'errore intercettato.

Soluzione:

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

struct rec {int volume; int costo; int quantita; struct rec *puntV; struct rec *puntC;};
struct rec *listaV, *listaC;

/* ListaC non serve in quanto si deve assumere che la lista sia vuota e quindi ListaC deve già essere NULL.LV si suppone che corrisponda a ListaV*/

struct rec *Read (struct rec *LV, char nome[], int *code)
{FILE   *f;  struct rec  *temp, *last;
  typedef struct {int volume; int costo; int quantita;} Trecord;      Trecord record;

  if ((f = fopen("a.dat", "rb"))==NULL)  {printf("errore apertura file");*code=0;return LV; }
 else
    { fread(&record, sizeof(Trecord), 1, f);
      while (feof(f)==0)
        { if ((temp= malloc(sizeof(struct rec)))==NULL)  {fclose (f);*code=0;return LV;}
          temp->volume = record.volume; temp->costo= record.costo; temp->quantita= record.quantita;
          temp->puntC=NULL; temp->puntV=NULL;

       //l'ordine del file coincide con quello della lista => inserimento di ogni record letto in coda
         if (LV==NULL) LV=temp;
         else { last=LV; while (last->puntV!= NULL) last=last->puntV;
                  last->puntV=temp;
                }
        fread(&record, sizeof(Trecord), 1, f);
        }
    }
 fclose (f);*code=1;return LV;
}
```

Esercizio 3

Si scriva il sottoprogramma OrdinaC che riceve come parametro la precedente lista dei prodotti e provvede a generare il collegamento tra gli elementi della lista in base ai valori di costo. Alla fine del sottoprogramma ListaC punterà al prodotto di costo maggiore e tutti i puntatori puntC saranno correttamente inizializzati per rispettare l'ordine stabilito nel precedente esercizio.

Soluzione

```
#include <stdio.h>
#include <stdlib.h>
struct rec {int volume; int costo; int quantita; struct rec *puntV; struct rec *puntC;};
struct rec *listaV, *listaC;
/* Si suppone che ListaC contenga Null come tutti i campi puntC degli elementi della lista - ipotesi fatta in aula */

struct rec *OrdinaC (struct rec *LV, struct rec *LC)
 struct rec  *lastinserted=NULL; //punta all'ultimo elemento immesso nella lista per costo
  struct rec  *temp; // per scandire la lista
  struct rec  *max; // per marcare il max in ogni fase di scansione
 int finito=0;

if (listaV==NULL)return listaC;

while (!finito)  //ad ogni iterazione inserisco 1 elemento nella lista per costo
{  temp=listaV; max=NULL;
   while(temp!=NULL)  //cerca max in set elementi non ancora sistemati
```

```
  { if( (temp!=lastinserted)&& (temp->puntC==NULL)) // caratteristiche dell'elemento non ancora sistemato
     if(max==NULL)max=temp; //carica come max il primo da sistemare
     else if(max->costo < temp->costo) max=temp; // cerca max tra gli altri
    temp=temp->puntV;
  }
  if (max==NULL)finito=1; //non ci sono più elementi da sistemare
  else {if (listaC==NULL) listaC=max;
        else  lastinserted->puntC=max;
        lastinserted=max;}
}
}
```

Esercizio 4. Solo per gli studenti di fondamenti di informatica

Dato il seguente programma

```
int d1,d2=1, status; pid_t id;
int main ()
{
 id=fork();
 if (id ==-1) {printf("\nan error occurred\n"); exit(-1);}
//istante T1

else
   {
    if (id == 0)
        {prima(d1);
//istante T2
         exit(1);
        }
    else
        { seconda(d1);
//istante T3
          id=wait(&status);
          d1=400;
//istante T4
         }
   }
}
void prima(int c) {c=4;}
void seconda (int c) {d2=111; c=100;}
```

Eseguire il programma e riportare nelle seguenti tabelle il valore delle variabili d1, d2 e status nei quattro istanti di tempo segnati col commento nel programma

Soluzione:

Premessa: l'esecuzione delle due funzioni (prima, seconda) non produce cambiamenti ai parametri in quanto passati per valore e quindi l'unica modifica visibile all'esterno è d2=111.

Processo Padre

	Istante T1	Istante T2	Istante T3	Istante T4
D1	? X(1) NE(2)	X	X	400
D2	? 1(1) NE(2)	? 1 o 111	111	111
Status	? X(1) NE(2)	X	X	256

Istante T1: situazione (1) se la fork ha avuto successo oppure (2) altrimenti

Istante T3: padre esiste e modifica solo d1

Istante T4: riceve status e modifica d1

Istante T2 nel figlio: padre esiste e si trova tra T1 e time di(wait) < T4 e quindi non posso sapere se d2 sia già stato modificato

Processo Figlio

	Istante T1	Istante T2	Istante T3	Istante T4
D1	? X(1) NE(2)	X	? X NE	NE
D2	? 1(1) NE(2)	1	? 1 NE	NE
Status	? X(1) NE(2)	X	? X NE	NE

Istante T1: situazione (1) se la fork ha avuto successo oppure (2) altrimenti

Istante T2: figlio esiste ma non cambia valori iniziali
Istante T4: figlio “killed” dato che siamo nel padre dopo la wait
Istante T3: entrambi i processi esistono e il figlio si trova dopo T1, ma potrebbe anche essere zombie

Usare i seguenti valori
NE: quando il processo non esiste;
X: se la variabile esiste ma non è stata ancora inizializzata;
?(): se non si è in grado di valutare il valore nell’istante di tempo richiesto; in questo caso indicare tra parentesi quali valori potrebbe assumere la variabile.