Fondamenti di Informatica L-A (A.A. 2004/2005) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di Mercoledì 13 Luglio 2005 – durata 2h30m

ESERCIZIO 1 (10 punti)

Un sistema per le previsioni meteorologiche utilizza un insieme di termometri siti in diverse località italiane. Ogni giorno, alle 02:00 e alle 15:00, un programma centrale recupera le temperature (un float compreso sempre tra -100.0 e 100.0) dai termometri e salva tali dati su due file di testo, separando i valori con uno spazio o con un "newline" ("**`minime.txt`**" per i valori notturni, "**`massime.txt`**" per i valori diurni). Si deve realizzare un programma che, leggendo tali dati dai due file, stampi a video il valore più basso e quello più alto sia delle minime che delle massime, e poi per ogni località stampi a video il valore notturno, il valore diurno e la media tra i due. In particolare, si realizzi:

1. una funzione **`readFromAFile(…)`** che riceva in ingresso il nome di un file di testo, e due parametri passati per riferimento, **`lowerValue`** e **`upperValue`**, di tipo **`float`**; la funzione deve restituire una lista contenente i valori presenti sul file specificato, e tramite i due parametri passati per riferimento deve restituire il minimo e il massimo dei valori letti. (5 punti)
2. Un programma **`main`** che, dopo aver richiesto all'utente di inserire i nomi dei due file, legga i valori registrati dai file (utilizzando due volte la funzione **`readFromAFile`**). Il programma quindi stampi a video la minima più bassa e la minima più alta, seguito poi dalla massima più bassa e dalla massima più alta; infine si stampi a video ogni valore di minima e di massima, seguito dalla media dei due valori. (5 punti)

Al fine di svolgere l'esercizio, il candidato consideri di avere a disposizione il tipo di dato astratto **`lista`** (già definito con elementi di tipo **`float`**), e le funzioni primitive relative, che pertanto possono non essere riportate nella soluzione.

ESERCIZIO 2 (12 punti)

È dato un file binario, di nome "**`valori.dat`**", contenente una sequenza di **`int`** separati da spazi o da caratteri di **`newline`**; non è noto a priori quanti interi siano presenti nel file. I valori sono disposti in ordine casuale. Si realizzi un programma che, letti dal file tali valori interi, li stampi a video ponendo prima i numeri pari e poi i numeri dispari. A tal scopo si definisca:

1. una funzione

```
int readLength(FILE *f, int *even, int *odd)
```

   che determini quanti valori sono presenti nel file. In particolare, la funzione deve restituire il numero totale di valori presenti nel file, e tramite i parametri **`even`** e **`odd`** deve restituire il numero di valori pari e di valori dispari rispettivamente (la somma di **`even`** + **`odd`** deve ovviamente essere uguale al numero totale di valori presenti nel file). (punti 5)

2. Un programma **`main`** che, aperto opportunamente il file "**`valori.dat`**", determini quanti valori sono presenti sul file tramite la funzione **`readLength(…)`**. Il programma deve allocare dinamicamente memoria sufficiente per leggere tutti i valori, e deve poi procedere a leggere i valori dal file e a disporli ordinatamente nel vettore allocato. Ad esempio, se nel file ci sono 13 valori pari e 16 valori dispari, nelle prime 13 posizioni del vettore ci dovranno essere i valori pari, e nelle seguenti 16 i valori dispari. Si ricorda al candidato l'esistenza della procedura di libreria **`void rewind(FILE *f)`** che riporta la testina di lettura a inizio file. Il programma stampi infine a video tale vettore. (punti 7)

ESERCIZIO 3 (5 punti)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione (si motivi opportunamente la risposta data)?

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 16

int pos = DIM-7;

void round(char str[], int pos) {
      int size, i;
      char temp;

      if (pos == 0)
            { pos=4; return; }
      else {
            for (size=0; str[size] != '\0'; size++);
            temp = str[0];
            for (i=0; i<size-1; i++) str[i] = str[i+1];
            str[i] = temp;
            round(str, pos-1);
      }
}

int main() {
      int i;
      char str[] = "StarWars";
      char * test;
      char temp;

      test = (char *) malloc(sizeof(char) * pos);
      test[pos-1] = str[pos-1];
      for (pos--; pos>0; pos--)
            test[pos-1] = str[pos-1];

      printf("%s\n", test);
      round(test, 1);
      printf("%s\n", test);
      printf("Pos vale: %d\n", pos);

      return 0;
}
```

ESERCIZIO 4 (5 punti)

Si illustri come avviene il passaggio dei parametri nell'invocazione di una funzione/procedura in linguaggio C. In particolare, si discuta la differenza nel passaggio di un parametro struct rispetto ad un parametro array.

SOLUZIONE

ESERCIZIO 1

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

#include "element.h"
#include "list.h"

#define DIM 256

list readFromAFile(char *fileName, float *lowerValue, float *upperValue) {
      FILE *source;
      list result = emptylist();
      float temp;
      *lowerValue = 100;
      *upperValue = -100;

      if ((source = fopen(fileName, "r")) == NULL) {
                  printf("Error opening the file %s\n", fileName);
                  exit(-1);}
      while (fscanf(source, "%f", &temp) != EOF) {
            result = cons(temp, result);
            if (temp < *lowerValue) *lowerValue = temp;
            if (temp > *upperValue) *upperValue = temp;
      }
      fclose(source);
      return result;
}

int main()
{
      char fileNameMinTemp[DIM];
      char fileNameMaxTemp[DIM];
      float lowerTempMin, upperTempMin, lowerTempMax, upperTempMax, avg;
      list minTemp, maxTemp;

      printf("Insert file name min temperatures: ");
      scanf("%s", fileNameMinTemp);
      printf("Insert file name max temperatures: ");
      scanf("%s", fileNameMaxTemp);

      minTemp = readFromAFile(fileNameMinTemp, &lowerTempMin, &upperTempMin);
      maxTemp = readFromAFile(fileNameMaxTemp, &lowerTempMax, &upperTempMax);

      while (!empty(minTemp) || !empty(maxTemp)) {
            avg = (head(minTemp) + head(maxTemp)) / 2;
            printf("Min Value: %f\n Max Value: %f\nAverage: %f\n\n",
                  head(minTemp), head(maxTemp), avg);
            minTemp = tail(minTemp);
            maxTemp = tail(maxTemp);
      }

      return 0;
}
```

ESERCIZIO 2

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

int readLength(FILE *f, int *even, int *odd) {
      int temp;
      *even = 0;
      *odd = 0;

      while(fread(&temp, sizeof(int), 1, f) == 1) {
            if ((temp%2) == 0) (*even)++;
            else (*odd)++;
      }
      return *even + *odd;
}

int main()
{
      FILE *f;
      int odd, even, lung;
      int *store;
      int i=0, j=0;
      int temp;

      if ((f = fopen("valori.dat", "r")) == NULL) {
            printf("Error opening the file %s\n", "valori.dat");
            exit(-1);
      }
      lung=readLength(f, &even, &odd);
      rewind(f);
      store = (int *) malloc(sizeof(int) * (lung));
      while(fread(&temp, sizeof(int), 1, f) == 1) {
            if (temp%2 == 0) {
                  store[i] = temp;
                  i++;
            }
            else {
                  store[even+j] = temp;
                  j++;
            }
      }
      fclose(f);
      for (i=0; i<(lung); i++)
            printf("%d ", store[i]);

      return 0;
}
```

ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:

StarWars
tarWarsS
Pos vale: 0

Nella fase iniziale del programma **`main`** vengono dichiarati un array di caratteri (istanziato con la stringa “StarWars”), un puntatore a carattere, un carattere **`temp`**, e un intero. Il primo gruppo di istruzioni alloca memoria sufficiente (riferita tramite il puntatore **`test`**) per copiare l’intera stringa “StarWars” (è aggiunto anche il terminatore al fine di ottenere una stringa ben formata). Nel secondo

gruppo di istruzioni, dopo aver stampato il contenuto della stringa riferita dal puntatore test, viene invocata la funzione **`round(…)`**, con parametri “StarWars” e **`pos`**=1.
La funzione **`round(…)`**, dopo aver determinato la lunghezza della stringa **`str`** ricevuta in ingresso, provvede a fare uno “shift” verso sinistra dei caratteri; i caratteri sono spostati di tante posizioni in base a quanto specificato tramite il parametro **`pos`**, e i valori che sono all’inizio della stringa vengono re-inseriti in coda alla stringa stessa, in ordine. Quindi la stringa **`test`** viene modificata nel seguente modo: la lettera ‘S’ viene inserita al termine della stringa, e tutte le altre lettere sono spostate di una posizione a sinistra (una sola posizione perché **`pos`** vale 1). La funzione **`round`** si reinvoca ricorsivamente, ma tale invocazione non causa nessun cambiamento (poiché **`pos`** assume valore 0 e la funzione termina immediatamente). Il **`main`** stampa a video il nuovo risultato e quindi il valore di **`pos`**, che è una variabile globale; tale variabile è stata modificata nel ciclo di copia precedente e alla fine di tale ciclo vale 0.