Laboratorio di Elementi di Architetture e Sistemi Operativi

Soluzioni del Compitino del 17 Aprile 2013

Esercizio 1. Un società organizzatrice di un concerto vuole verificare che non ci siano biglietti falsi. Scrivere un programma in C che acquisisca i numeri di serie dei biglietti e verifichi che non vi siano numeri duplicati:

- Il programma acquisisce innanzitutto il numero di biglietti venduti, `N`, ed in seguito acquisisce i numeri di serie degli `N` biglietti.
- Al termine dell'acquisizione, il programma stamperà "`Tutto regolare`" se non si sono riscontrati duplicati, altrimenti stamperà il numero di serie dei biglietti duplicati.

Utilizzare le funzioni di allocazione dinamica della memoria per permettere al programma di funzionare per qualsiasi valore di `N`. Strutturare il codice scomponendo il problema in operazioni semplici, e scrivendo una funzione apposita per ognuna di esse.

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

int *crea(int n) {
  int *v;
  v = (int *)malloc(n*sizeof(int));
  return v;
}

void popola(int *v, int n) {
  int i;
  printf("Inserire i numeri di serie dei biglietti:\n");
  for(i = 0; i < n; i++) {
    scanf("%d", v+i);
  }
}

int duplicati(int *v, int n) {
  int i, j, flag=0;

  for(i = 0; i < n; i++) {
    for(j = i+1; j < n; j++) {
      if(v[i] == v[j]) {
        printf("Il biglietto %d e' duplicato!\n", v[i]);
        flag = 1;
      }
    }
  }
  if(flag == 0) {
    printf("Tutto regolare!\n");
  }
  return flag;
}
```

```
int main() {
  int n;
  int *v;

  printf("Inserire il numero di biglietti da controllare: ");
  scanf("%d", &n);

  if(n <= 0) {
    printf("Il numero di biglietti deve essere positivo!\n");
    return 2;
  }

  v = crea(n);
  if(v == NULL) {
    printf("impossibile allocare la memoria!\n");
    return 3;
  }
  popola(v, n);
  return duplicati(v, n);
}
```

Esercizio 2. Scrivere un programma C per moltiplicare matrici quadrate di interi. Il programma deve:

- prendere come *parametri della linea di comando* due nomi di file;
- leggere la dimensione `N` ed i valori di due matrici `N` x `N` dal primo file;
- calcolare il *prodotto righe per colonne* delle due matrici, secondo l'espressione

$$c_{ij} = \sum_{k=0}^{N-1} a_{ik} b_{kj} = a_{i0} b_{0j} + a_{i1} b_{1j} + \cdots + a_{iN-1} b_{N-1j}$$

- salvare il risultato sul secondo file.

Strutturare il codice scomponendo il problema in operazioni semplici, e scrivendo una funzione apposita per ognuna di esse. Utilizzare le funzioni di allocazione dinamica della memoria per permettere al programma di funzionare per qualsiasi valore di `N`. Si supponga che il file di input sia strutturato nel modo seguente:

- il primo valore numerico rappresenta la dimensione `N` delle matrici;
- successivamente, sono elencati i valori della prima matrice ordinati per righe, seguiti dai valori della seconda matrice (sempre elencati per righe).

Stampare l'output utilizzando lo stesso formato dell'input: prima la dimensione della matrice, seguita dai valori elencati per righe.

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

void printerror(char *str, int n) {
  fprintf(stderr, "Errore %d: %s\n", n, str);
  exit(n);
}
```

```
int *crea(int n) {
  int *pm;
  pm = malloc(n*n*sizeof(int));
  if(pm == NULL)
    printerror("impossibile allocare la memoria!", 5);
  return pm;
}

void popola(FILE *fin, int *m, int n) {
  int i, res;
  for(i = 0; i < n*n; i++) {
    res = fscanf(fin, "%d", m+i);
    if(res == EOF)
      printerror("impossibile leggere il file", 2);
    if(res != 1)
      printerror("file di input non formattato correttamente", 3);
  }
}

void scrivi(FILE *fout, int *m, int n) {
  int i, j, res;

  if(fprintf(fout, "%d\n", n) == EOF)
    printerror("Errore nella scrittura del file", 4);

  for(i = 0; i < n; i++) {
    for(j = 0; j < n; j++) {
      if(fprintf(fout, "%d\t", *(m+i*n+j)) == EOF) {
        printerror("Errore nella scrittura del file.", 4);
      }
    }
    if(putc('\n',fout) == EOF)
      printerror("Errore nella scrittura del file.", 4);
  }
}

void moltiplica(int *ris, int *a, int *b, int n)
{
  int i, j, k;
  for(i = 0; i < n; i++) {
    for(j = 0; j < n; j++) {
      ris[i*n+j] = 0;
      for(k = 0; k < n; k++) {
        ris[i*n+j] += a[i*n + k] * b[k*n + j];
      }
    }
  }
}
```

```
int main(int argc, char *argv[]) {
  FILE *fin, *fout;
  int res, n;
  int *m1, *m2, *m3;

  if (argc != 3)
    printerror("Specificare due file come argomenti", 1);
  fin = fopen(argv[1],"r");
  if (fin == NULL)
    printerror("Errore nell'apertura del primo file.", 2);
  fout = fopen(argv[2],"w");
  if (fout == NULL)
    printerror("Errore nell'apertura del secondo file.", 2);


  res = fscanf(fin, "%d", &n);
  if (res == EOF)
    printerror("Errore nella lettura del file", 2);
  if(res != 1)
    printerror("Input in formato non corretto.", 3);

  m1 = crea(n);
  popola(fin, m1, n);
  m2 = crea(n);
  popola(fin, m2, n);

  m3 = crea(n);
  moltiplica(m3, m1, m2, n);
  scrivi(fout, m3, n);

  if ( fclose(fin) != 0 ) {
    printerror("Errore nella chiusura del primo file.", 5);
  }
  if ( fclose(fout) != 0 ) {
    printerror("Errore nella chiusura del secondo file.", 5);
  }
  return 0;
}
```