Problem solving elementare su dati vettoriali

Esercizi risolti

1 Esercizio: “Prodotto di matrici”

Si scriva un programma che, lette da tastiera le dimensioni di due matrici di interi e acquisite le due matrici, ne calcoli la matrice prodotto e la visualizzi.

Si ricorda che, date due matrici A (di dimensione m * n) e B (di dimensione (n * p), si definisce *prodotto* tra le matrici A e B la matrice C = A*B, il cui generico elemento $c_{i,j}$ è la somma dei prodotti degli elementi della i-esima riga di A per i corrispondenti elementi della j-esima colonna di B, ovvero

$$c_{ij} = \sum_{k=1}^{n} a_{ik}\ b_{kj}$$

Soluzione

Al fine di evitare ripetizioni e migliorare la modularità del programma, si presenta una soluzione con funzioni per input di dimensioni e matrici, di verifica delle dimensioni, calcolo del prodotto e visualizzazione del risultato.

```
#include <stdio.h>

#define MAX 10

/* dichiarazione funzioni (prototipi) */
void leggi_dim (int *rp, int *cp);
int verifica_dim (int r, int c);
void leggi_mat (float mat[MAX][MAX], int r, int c);
void prodotto (float m1[MAX][MAX], int r1, int c1,
               float m2[MAX][MAX], int r2, int c2,
               float m3[MAX][MAX], int r3, int c3);
void stampa (float m1[MAX][MAX], int c1,
             float m2[MAX][MAX],
             float m3[MAX][MAX], int r3, int c3);

/* programma principale */
void main (void) {
  float m1[MAX][MAX], m2[MAX][MAX], m3[MAX][MAX];
  int r1, c1, r2, c2, r3, c3;

  printf("*** Prodotto di matrici ***\n");
  leggi_dim (&r1, &c1);
  leggi_dim (&r2, &c2);

  if (verifica_dim (c1, r2) == 1)
    printf ("Matrici non compatibili\n");
  else {
    leggi_mat (m1, r1, c1);
    leggi_mat (m2, r2, c2);
    r3 = r1;
    c3 = c2;
    prodotto (m1, c1, m2, m3, r3, c3);
    stampa (m1, r1, c1, m2, r2, c2, m3, r3, c3);
  }
}
```



```
/* lettura dimensioni matrice */
void leggi_dim (int *rp, int *cp) {
  do {
    printf ("Numero righe matrice: "); scanf ("%d", rp);
    printf ("Numero colonne matrice: "); scanf ("%d", cp);
  } while (*rp > MAX || *cp > MAX);
}

/* verifica dimensione matrici */
int verifica_dim (int c1, int r2) {
  if (c1==r2) return (0);
  else return (1);
}

/* lettura matrici */
void leggi_mat (float mat[MAX][MAX], int r, int c) {
  int i, j;

  /* lettura matrice */
  for (i=0; i<r; i++)
    for (j=0; j<c; j++) {
      printf ("m[%d,%d] = ", i, j);
      scanf ("%f", &mat[i][j]);
    }
}

/* esecuzione del prodotto */
void prodotto (float m1[MAX][MAX], int c1,
               float m2[MAX][MAX],
               float m3[MAX][MAX], int r3, int c3)
{
  int i, j, k;

  for (i=0; i<r3; i++)
    for (j=0; j<c3; j++) {
      m3[i][j] = 0;
      for (k=0; k<c1; k++)
        m3[i][j] += m1[i][k] * m2[k][j];
    }
}

/* visualizzazione del risultato */
void stampa (float m1[MAX][MAX], int r1, int c1,
             float m2[MAX][MAX], int r2, int c2,
             float m3[MAX][MAX], int r3, int c3) {
  int i, j;

  for (i=0; (i<r1 || i<r2); i++) {
    if (i < r1)
      for (j=0; j<c1; j++)
         printf ("%7.2f ", m1[i][j]);
    if (i == (r1/2)) printf ("  *  ");
    else printf ("     ");
    if (i < r2)
      for (j=0; j<c2; j++)
         printf ("%7.2f ", m2[i][j]);
    if (i == (r1/2)) printf ("  =  ");
    else printf ("     ");
    if (i < r1)
      for (j=0; j<c3; j++)
         printf ("%7.2f ", m3[i][j]);
    printf ("\n");
  }
}
```

Osservazione

Si osservi il formato di visualizzazione utilizzato attraverso la presentazione di un esempio di esecuzione:

```
*** Prodotto di matrici ***
Numero righe matrice: 2
Numero colonne matrice: 3
Numero righe matrice: 3
Numero colonne matrice: 2
m[1,1] = 1
m[1,2] = 2
m[1,3] = 3
m[2,1] = 21
m[2,2] = 32
m[2,3] = 43
m[1,1] = -6
m[1,2] = -7
m[2,1] = -8
m[2,2] = 3
m[3,1] = 4
m[3,2] = 5.6
 1.00   2.00   3.00        -6.00   -7.00         -10.00   15.80
21.00  32.00  43.00    *   -8.00    3.00     =  -210.00  189.80

                            4.00    5.60
```

2 Esercizio: “Somma in base B”

Si vuole realizzare una funzione in grado di effettuare somme per numeri interi rappresentati in una base B (compresa tra 2 e 10). La funzione, ricevuti gli operandi su 2 stringhe di caratteri, deve operare la somma calcolando le somme parziali sulle singole cifre e propagando i riporti, generando il risultato su una terza stringa, allocata dinamicamente e ritornata come puntatore a carattere.

Soluzione

```
char *sommaBase(char *x, char *y, int B) {
  /* lunghezze (numero ci cifre) degli operandi */
  int lx=strlen(x), ly=strlen(y);
  /* la lunghezza del risultato (ls) e' <= al massimo tra lx e ly,
     + 1 per eventuale riporto uscente da somma cifre piu' significative */
  int ls = (lx>ly?lx:ly)+1;
  int i, xInt, yInt, sInt, riporto;

  /* stringa risultato */
  char *s = malloc((ls+1) * sizeof(char));

  riporto = 0; s[ls] = '\0';
  for (i=1; i<ls; i++) {
    /* leggi cifre i-esime (da destra) di x e y, converti da char a int */
    xInt = (i>=lx)? 0: x[lx-i]-'0';
    yInt = (i>=ly)? 0: y[ly-i]-'0';
    sInt = xInt + yInt + riporto;
    /* calcola riporto e cifra i-esima del risultato */
    riporto = sInt / B; sInt = sInt % B;
    /* converti da int a char */
    s[ls-i] = sInt+'0';
  }
  s[0] = riporto ? '1': ' ';

  Return (s);
}
```

3 Esercizio: “Cruciverba”

Una matrice bidimensionale di dimensioni non note a priori (tuttavia con un massimo di 20 righe e 20 colonne) rappresenta lo schema di un cruciverba. Gli elementi della matrice, che possono assumere solamente i valori 0 e 1, rappresentano le caselle: agli elementi di valore 0 corrispondono le caselle bianche del cruciverba, mentre agli elementi di valore 1 corrispondono le caselle nere.

Realizzare un programma che:

- legga da tastiera il nome del file contenente la matrice
- legga da tale file la matrice indicata
- visualizzi il cruciverba corrispondente con il formato indicato nell’esempio seguente e con una appropriata numerazione delle caselle.

Si tenga presente che una casella viene numerata se è bianca e:

- la casella sottostante è bianca e non vi sono caselle bianche al di sopra

oppure

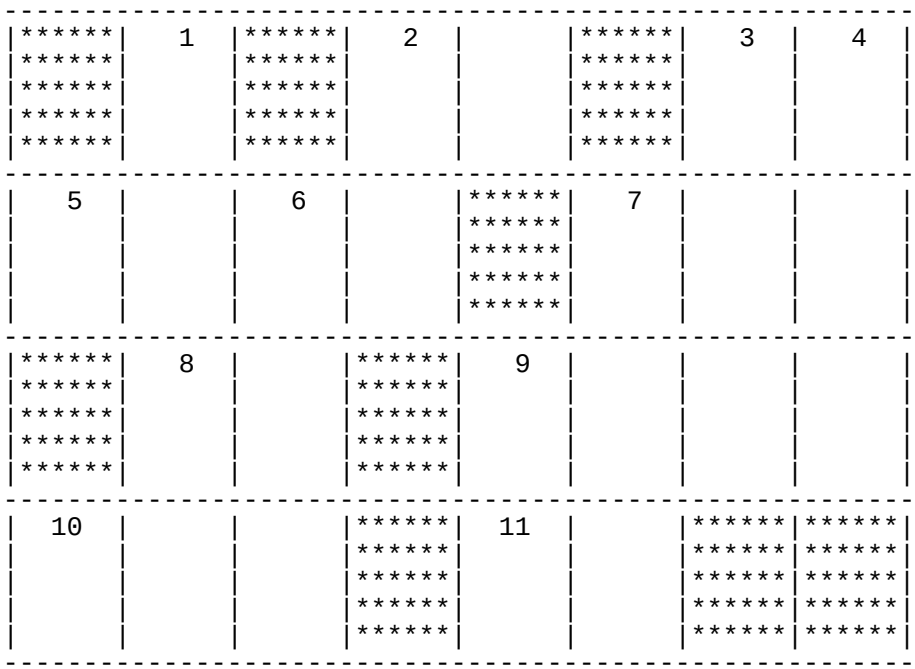
- la casella sulla destra è bianca e non vi sono caselle bianche a sinistra.

Esempio

Si supponga che il file di ingresso sia il seguente:

```
10100100
00001000
10010000
00010011
```

L’esecuzione del programma deve produrre, su video, il seguente cruciverba:



Soluzione

```
#include <stdio.h>

/* definizioni costanti */
#define DIM  20
#define L    30
/* Numero di Righe Visualizzate per ogni cella: */
#define NRV   5
/* Numero di Colonne Visualizzate per ogni cella: */
#define NCV   6
#define TRUE  1
#define FALSE 0

/* prototipi */
int lettura (char mat[DIM][DIM], int *nr, int *nc);
void riga (int nc);
void piena (void);
void vuota (void);
void vuota_numerata (int *contatore);
int numeri (char mat[DIM][DIM], int nr, int nc, int i, int j);

/* programma principale */
void main (void)
{
  char mat[DIM][DIM];
  int i, j, rip, nr, nc, contatore;

  if (lettura (mat, &nr, &nc) == TRUE) {
    contatore = 1;
    for (i=0; i<nr; i++) {
      riga ('-',nc*(NCV+1)+1,-1,"\n"); /* riga di trattini */
      /* gestione della i-esima riga della matrice */
      for (rip=0; rip<NRV; rip++) {
        for (j=0; j<nc; j++) {
          /* gestione elemento i, j */
          printf ("|");
          if (mat[i][j] == '1')
            riga ('*',NCV,-1,"");
          else if (mat[i][j] == '0') {
            /* controllo sulla numerazione */
            if (rip == 0 && numeri (mat, nr, nc, i, j) == TRUE)
              /* occorre numerare la casella */
              riga (' ',NCV,contatore++,"");
            else
              riga (' ',NCV,-1,"");
          }
          else
            /* errore */
            exit (1);
        } /* for j */
        printf ("|\n");
      } /* for rip */
    } /* for i */
    riga ('-',nc*(NCV+1)+1,-1,"\n"); /* riga di trattini */
  }
}
```

```
/* lettura da file con caricamento matrice e
   determinazione della sua dimensione */
int lettura (char mat[DIM][DIM], int *nr, int *nc) {
  char nome[L];
  char riga_file[DIM+2];
  int i, j;
  FILE *fp;

  printf ("Input nome file: ");
  scanf ("%s", nome);
  fp = fopen (nome, "r");
  if (fp == NULL) {
    printf ("FILE NON APRIBILE\n");
    return (FALSE);
  }
  else {
    /* lettura file con memorizzazione in una matrice */
    for (i=0; fgets(riga_file,DIM+1,fp)!=NULL; i++) {
      for (j=0; riga_file[j]!='\n'; j++) {
        mat[i][j] = riga_file[j];
      }
    }
    fclose (fp);

    *nc = j;
    *nr = i;
    return (TRUE);
  }
}

/* visualizzazione di una riga ---- ... */
void riga (char c, int nc, int num, char *s1) {
  int i;

  for (i=0; i<nc/2-1; i++) printf ("%c",c);
  if (num>=0) printf ("%2d", num);
  else printf ("%c%c",c,c);
  for (i=0; i<nc/2-1; i++) printf ("%c",c);
  printf ("%s",s1);
}

/* determinazione della presenza della numerazione in una casella */
int numeri (char mat[DIM][DIM], int nr, int nc, int i, int j) {
  /* test sul numero "verticale" */
  if (i==0 || i<(nr-1)&&mat[i-1][j]!='0') {
    if (mat[i+1][j]=='0')
      return TRUE;
  }
  /* test sul numero "orizzontale" */
  if (j==0 || j<(nc-1)&&mat[i][j-1]!='0') {
    if (mat[i][j+1] == '0' )
      return TRUE;
  }

  /* ne' numerazione "verticale" ne' numerazione "orizzontale" */
  return FALSE;
}
```

4 Esercizio: “Eliminazione di spazi”

Scrivere un programma che:

- legga un testo composto da un numero imprecisato di righe da un file di nome testoin
- compatti tutti gli spazi multipli in spazi singoli
- aggiunga la metà (arrotondata all’intero inferiore) degli spazi eliminati all’inizio della riga
- scriva il risultato su un secondo file di nome testoout.

Il programma deve comunque mantenere la suddivisione in righe presente nel file di input.

Esempio

Se si effettua la lettura del seguente testo:

```
Il programma  lavora   in questo  modo !
```

il formato di uscita sarà il seguente:

```
 Il programma lavora in questo modo !
```

Soluzione

Risulta evidente come si possa operare riga per riga, con la necessità di memorizzazione ogni riga in un vettore di caratteri (in modo da conteggiare gli spazi) prima di iniziare l’output della riga:

1. si aprono i due file
2. si legge una riga del primo file e la si memorizza in una stringa di caratteri
3. si manipola tale stringa eliminando e aggiungendo gli spazi e scrivendo poi tale stringa su testoout
4. si ripete dal punto 2 sino a incontrare la fine di testoin.

```
#include <stdio.h>

/* lunghezza massima delle righe di testoin */
#define MAX 80

void main (void)
{
  FILE *fp1, *fp2;
  char str[MAX+1];
  int i, j;

  /* apertura file */
  fp1 = fopen ("testoin", "r");
  fp2 = fopen ("testoout", "w");

  /* controllo operazione */
  if (fp1 == NULL || fp2 == NULL)
    printf ("ERRORE NELL'APERTURA FILE ! \n");
  else {
    while (fgets(str,MAX,fp1) != NULL ) {
      i=0;
      for (i=j=0; str[i]!='\0'; i++) {
        if (i=0 || str[i]!=' ' || str[i-1]!=' ')
          str[j++]=str[i];
      }
      for (i=(i-j)/2; i>=0; i--) fprintf (fp2, " ");
      fprintf (fp2, "%s\n", str);
    }
```

```
    fclose (fp1);
    fclose (fp2);
  }
}
```

5 Esercizio: “Eliminazione dei valori nulli”

Si scriva un programma che:

- legga una sequenza di numeri interi di lunghezza MAX
- elimini da tale sequenza tutti i valori nulli (compattando i rimanenti) e stampi la sequenza così ottenuta in ordine inverso.

Esempio

Nel caso in cui MAX sia pari a 16 e la sequenza letta sia la seguente:

```
15  32  37   9   0  67   3  -2   0   8  10  -1   0   0  34   4
```

La sequenza visualizzata è la seguente:

```
4  34  -1   10  8   -2  3   67  9  37  32   15
```

Soluzione

```
#include <stdio.h>

#define   MAX   16

void main (void)
{
  int vet[MAX];
  int i;

  /* lettura */
  printf ("SEQUENZA DA COMPATTARE: ");
  for (i=0; i<MAX; i++)
    scanf("%d", &vet[i]);

  /* compattazione e stampa */
  for (i=MAX-1; i>=0; i--)
    if (vet[i]!=0)
      printf ("%d ", vet[i]);
  printf ("\n");
}
```

Si osservi come sia necessario un vettore di appoggio, in quanto l’ordine dei dati va invertito.

6 Esercizio: “Ordinamento di un vettore tramite scambio”

Si legga da tastiera un vettore di MAX interi e si effettui su di esso un ordinamento in ordine crescente mediante il seguente algoritmo (noto come *ordinamento per scambio*, *exchange sort* o *bubble sort*):

1. si pone i=0
2. si pone j=0
3. si considerano gli elementi j e j+1 e, qualora essi non siano nell’ordine desiderato, li si scambia
4. si incrementa j di una unità e si ripete il passo precedente sino alla coppia MAX-i-1, MAX-i (nel caso di i=0 questi sono gli ultimi due elementi del vettore)
5. si incrementa i di una unità
6. si ripetono i passi dal passo 2 in poi sino a quando i=MAX-2, condizione che garantisce l’ordinamento del vettore.

In pratica l’algoritmo simula l’effetto di una bolla che all’interno di un liquido sale verso l’alto: a ogni iterazione, attraverso confronti di elementi contigui, l’elemento maggiore sale *verso l’alto (parte finale del vettore)*, raggiungendo la posizione corretta. Ad ogni ogni iterazione, la parte finale del vettore è quella già ordinata, mentre quella iniziale contiene i dati ancora da ordinare.

Esempio

```
Vettore iniziale:
vet[0] = 8
vet[1] = 7
vet[2] = 4
vet[3] = 9
vet[4] = 1

Iterazione numero 1 : 7 4 8 1 9
Iterazione numero 2 : 4 7 1 8 9
Iterazione numero 3 : 4 1 7 8 9
Iterazione numero 4 : 1 4 7 8 9

RISULTATO :
vet[0] = 1
vet[1] = 4
vet[2] = 7
vet[3] = 8
vet[4] = 9
```

Esempio

```
Vettore iniziale:
vet[0] = 5
vet[1] = 4
vet[2] = 3
vet[3] = 2
vet[4] = 1

Iterazione numero 1: 4 3 2 1 5
Iterazione numero 2: 3 2 1 4 5
Iterazione numero 3: 2 1 3 4 5
Iterazione numero 4: 1 2 3 4 5
Iterazione numero 5: 1 2 3 4 5

RISULTATO :
vet[0] = 1
vet[1] = 2
vet[2] = 3
vet[3] = 4
vet[4] = 5
```


Soluzione

```
#include <stdio.h>

#define MAX 5

void main (void)
{
  int i, j, vetvar, vet[MAX];

  /* ciclo di lettura vettore */
  printf("Vettore iniziale:\n");
  for (i=0; i<MAX; i++) {
    printf ("vet[%d] = ", i); scanf ("%d", &vet[i]);
  }
  printf("\n");

  /* ordinamento */
  for (i=0; i<MAX-1; i++) {
    for (j=0; j<MAX-1-i; j++) {
      if (vet[j] > vet[j+1]) {
        vetvar = vet[j];
        vet[j] = vet[j+1];
        vet[j+1] = vetvar;
      }
    }

  /*   stampa vettore dopo ogni iterazione principale */
  printf("Iterazione numero %d:\n", i+1);
  for (j=0; j<MAX; j++)
    printf ("%d", vet[j]);
  printf ("\n");
}

/* stampa vettore risultante */
printf("\nRISULTATO :\n");
for (i=0; i<MAX; i++)
   printf ("vet[%d] = %d\n", i, vet[i]);
}
```

Osservazioni

Gli algoritmi di ordinamento analizzati sino a ora non sono i più veloci e sono senza dubbio migliorabili dal punto di vista dell'efficienza. Per esempio, se l'algoritmo di scambio non effettua alcuno scambio durante una iterazione, significa che il vettore è già ordinato e si possono evitare le successive iterazioni.

Una modifica di tale genere è relativamente poco costosa e può essere realizzata introducendo una variabile apposita che memorizza il fatto che ci siano stati scambi o meno nel corso dell'ultima scansione del vettore. L'implementazione relativa viene di seguito riportata.

```
#include <stdio.h>

#define MAX 5

void main (main) {
  int flag;
  int i, j, vetvar, vet[MAX];

  /* ciclo di lettura vettore */
  printf("Vettore iniziale:");
  for (i=0; i<MAX; i++) {
    printf ("\nvet[%d] = ", i);
    scanf ("%d", &vet[i]);
  }
```

```
  printf ("\n");

  /* ordinamento */
  flag = 1;
  for (i=0; (i<MAX-1)&&(flag==1); i++) {
    flag = 0;
    for (j=0; j<MAX-1-i; j++) {
      if (vet[j] > vet[j+1]) {
        flag = 1;
        vetvar = vet[j];
        vet[j] = vet[j+1];
        vet[j+1] = vetvar;
      }
    }

    /*   stampa vettore   */
    printf ("Iterazione numero %d:\n", i+1);
    for (j=0; j<MAX; j++)
      printf ("%d", vet[j]);
    printf ("\n");
  }

  /* stampa vettore risultato  */
  printf ("\nRISULTATO :\n");
  for (i=0; i<MAX; i++)
    printf ("vet[%d] = %d\n", i, vet[i]);
}
```

Nel caso in cui il vettore sia già inizialmente ordinato, una sola iterazione è sufficiente a terminare l'algoritmo, come mostrato dal seguente esempio di esecuzione:

```
INTRODUCI IL VETTORE:
vet[0] = 1
vet[1] = 2
vet[2] = 3
vet[3] = 4
vet[4] = 5

Iterazione numero 1: 1 2 3 4 5

RISULTATO 1:
vet[0] = 1
vet[1] = 2
vet[2] = 3
vet[3] = 4
vet[4] = 5
```

