Esercizi Aggiuntivi su Matrici

Credits Prof. Campi

Esercizio

- Scrivere un programma che chiede all'utente di riempire una matrice, la stampa, cerca, se esiste, la prima occorrenza dello 0 e dice in che posizione è stata trovata

```
#include <stdio.h>
#define N 3
#define M 4
int main() {
	int i, j, si=0, m[N][M];
	for (i = 0; i < N; i++) {
	    for (j = 0; j < M; j++) {
	       printf("\nInserisci un elemento della matrice: "); scanf("%d",&m[i][j]);
	    }
	}
	printf("\nLa matrice inserita e': \n");
	for (i = 0; i < N; i++) {
	    for (j = 0; j < M; j++)
	       printf(" %d ", m[i][j]);
	    printf("\n");
	}
	/* Cerca lo 0 */
	for (i = 0; i < N && si==0; i++) {
	    for (j = 0; j < M && si==0; j++)
	       if (m[i][j]==0) { si=1; printf ("\n trovato lo 0 in riga %d e colonna %d",i,j); }
	}
	if (si==0) printf("\nLo 0 non e' stato trovato");
	return 0;
}
```

Esercizio

- Scrivere un programma che chiede all'utente di riempire una matrice, la stampa, cerca, se esiste, la prima occorrenza dello 0, l'ultima occorrenza dello 0 e l'occorrenza dello 0 in posizione mediana e dice in che posizione sono state trovate .

```
#include <stdio.h>
#define N 3
#define M 4
int main() {
    int m[N][N],i,j,trovato,cont=0,cont2=0;
    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            printf("\nInserisci un elemento della matrice: "); scanf("%d",&m[i][j]);
        }
    }
    printf("\nLa matrice inserita e': \n");
    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++)
            printf(" %d ", m[i][j]);
        printf("\n");
    }
    /* Cerca lo 0 */
    //prima
    …
    //ultima
    …
    //mediana
    …
    return 0;
}
```

```
//prima
trovato =0;
for(i=0;i<N && trovato==0;i++)
 for(j=0;j<M && trovato==0;j++)
  if(m[i][j]==0){
    trovato=1;
    printf ("primo in %d  %d",i,j);
  }

//ultima
 trovato=0;
 for(i=N-1;i>=0 && trovato==0;i--)
  for(j=M-1;j>=0 && trovato==0;j--)
   if(m[i][j]==0){
     trovato=1;
     printf ("ultimo in %d %d",i,j);
   }
```

```
//mediana
for(i=0;i<N;i++)
 for(j=0;j<M;j++)
  if(m[i][j]==0){
    cont++;
  }

if(cont%2==0)
 cont=cont/2;
else
 cont=cont/2+1;

for(i=0;i<N;i++)
 for(j=0;j<M;j++)
  if(m[i][j]==0){
    cont2++;
    if(cont2==cont)
      printf("%d %d ",i,j);
  }
```

Esercizio

- Scrivere un programma che riempia una matrice 20x30 chiedendo all'utente di inserire gli elementi, ma inserendo nella matrice solo gli elementi pari.
- Il programma termina quando la matrice è piena.

```
#define N 20
#define M 30
int main () {
        int i=0,j=0,n,mat[N][M];
        while(i<N) {
                j=0;
                while(j<M) {
                        scanf("%d",&n);

                        if(n%2==0) {
                                mat[i][j]=n;
                                j++
                        }
                }
                i++;
        }
        return 0;
}
```

```
#define N 20
#define M 30

int main () {
        int i=0,j=0,n,mat[N][M];

        for(i=0; i<N; i++){
                for(j=0; j<M; j++){
                        scanf("%d",&n);

                        if(n%2==0)
                                mat[i][j]=n;
                        else
                                j--;
                }
        }
        return 0;
}
```

Esercizio

- **Scrivere un programma C che legge una sequenza di numeri interi e li mette nella prima riga della matrice M. La lettura della sequenza termina quando alla prima riga della matrice M sono stati assegnati 50 interi oppure quando viene letto il secondo numero intero negativo.**

```
#include<stdio.h>
#define N 50
int main() {
    int mat[N][N],i=0,j,neg=0;

    do {
        scanf("%d",&mat[0][i]);
        if (mat[0][i]<0)
            neg++;
        i++;
     } while (neg < 2 && i<50);

     return 0;
}
```

Esercizio

- Scrivere un programma che chiede all'utente di inserire una matrice 20x30, poi (dopo aver terminato la fase di inserimento) copia gli elementi dispari in una seconda matrice 20x30 senza lasciare buchi, se non in fondo.
- Gli elementi in fondo (i "buchi") siano messi a zero.

```
#define N 20
#define M 30
int main () {
        int i=0,j=0,k=0,r=0,mat1[N][M],mat2[N][M]={0};
        for(i=0; i<N; i++)
                for(j=0; j<M; j++)
                        scanf("%d",&mat1[i][j]);
        for(i=0; i<N; i++) {
                for(j=0; j<M; j++) {
                        if(mat1[i][j]%2!=0) {
                                mat2[r][k]=mat1[i][j];
                                k++;
                                if(k==M) { k=0; r++; }
                        }
                }
        }
        return 0;
}
```

Esercizio (tde 14-11-2008)

- Scrivere un programma che chiede all'utente di inserire una matrice NxN con elementi tutti diversi. Se l'utente inserisce un numero già inserito il programma lo avvisa dell'errore e chiede nuovamente di inserire l'elemento.

```
#include <stdio.h>
#define N 10
int main(){
   int i,j,k,t,cont,A[N][N],howMany=0,ok=1;
   for (i=0;i<N;i++) {
            for (j=0;j<N;j++) {
                     do {
                              cont=0;
                              ok=1;
                              printf("Inserire un valore\n");
                              scanf("%d", &A[i][j]);
                              for (k=0;k<N && cont < howMany;k++) {
                                  for (t=0;t<N && cont < howMany;t++) {
                                          if (A[i][j] == A[k][t]) {
                                                  ok=0;
                                                  printf("Errore:valore duplicato\n");
                                          }
                                          cont++;
                                  }
                              }
                     } while (ok==0);
                     howMany++;
            }
   }
}
```

Esercizio (tde 13-7-2016)

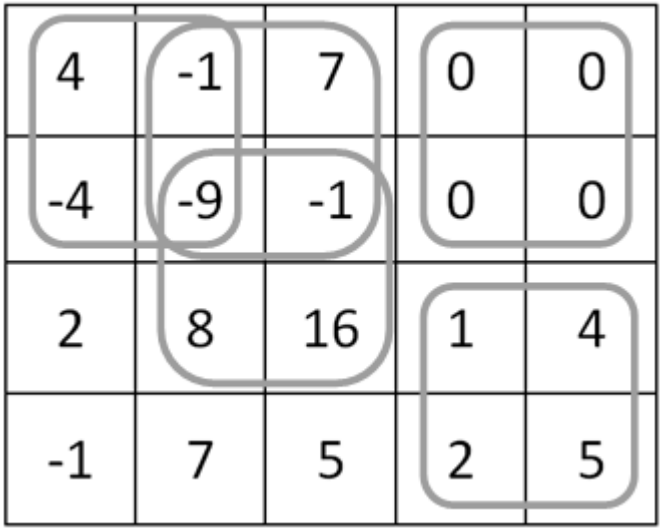
- Una matrice quadrata Mat di dimensioni NxN (con N costante predefinita) è diagonalmente dominante se la somma dei valori assoluti degli elementi su ciascuna riga, escluso l'elemento sulla diagonale principale, è minore del valore assoluto dell'elemento corrispondente sulla diagonale principale.
- Scrivere un programma che chiede all'utente di inserire i valori di una matrice e stampa «Dominante» se la matrice è diagonalmente dominante, «Non dominante» altrimenti.
- Si ricorda che la funzione int abs(int n) restituisce il valore assoluto dell'intero n ricevuto come parametro.

```
#include <stdio.h>
#define N 10
int main(){
   int mat[N][N],diag,ele,dom=1;
   for(i=0;i<r;i++)
      for(j=0;j<r;j++){
         printf("inserire il valore alla riga %d ed alla colonna %d : ",i+1,j+1);
         scanf("%d",&mat[i][j]);
      }

   for(i=0;i<N && dom;i++){
      diag=0; ele=0;
      for(j=0;j<N;j++)
         if(j==i)
            diag=abs(mat[i][j]);
         else
            ele+=abs(mat[i][j]);
      if(diag<=ele)
         dom=0;
   }
   if(dom)
       printf("Dominante\n");
   else
      printf("Non dominante\n");
}
```

Esercizio (tdeB 20-7-2010)

- Considerata una matrice A di N x M interi, definiamo *claque* una sottomatrice 2 x 2 in cui la somma algebrica dei valori di una diagonale sia pari a quella dell'altra diagonale. In figura sono evidenziate le claque.
- Si scriva un programma che acquisisce una matrice N x M stampa il numero di claque della matrice.



```
int main() {
  int m[N][M], i, j, cont = 0;

  printf("Inserisci %d elementi\n",N*M);

  for ( i = 0; i < N; i++ )
    for ( j = 0; j < M; j++ )
        scanf("%d", &m[i][j]);

  for ( i = 0; i < N-1; i++ )
    for ( j = 0; j < M-1; j++ )
      if(m[i][j] - m[i+1][j] - m[i][j+1] + m[i+1][j+1] == 0)
        cont++;

  printf("Le claque sono %d\n",cont);

  return 0;
}
```

Esercizio

- Si scriva un programma in linguaggio C che stampi sullo standard output il contenuto di un quadrato magico di dimensione n, con n dispari. Un quadrato magico di ordine n contiene i primi n numeri naturali (1, 2, 3, …., n^2) disposti in modo tale che la somma dei numeri su ogni riga, su ogni colonna e sulle due diagonali principali sia sempre la stessa.
- Es: n = 3

492	000	002	002	402	402	402	402	402	492
357	000	000	300	300	350	350	357	357	357
816	010	010	010	010	010	016	016	816	816

- Esiste una regola molto semplice per percorrere la matrice disponendo i numeri interi in ordine crescente. Partendo col posizionare un 1 nella posizione centrale sull'ultima riga, si percorre la matrice incrementando di una unità il numero di riga e il numero di colonna dell'elemento attuale, avendo cura di considerare i bordi opposti della matrice come adiacenti. Se durante questa operazione si individua una cella vuota si scrive il numero successivo; altrimenti, il numero successivo, viene posizionato nella cella avente riga immediatamente superiore a quella dell'ultimo numero inserito.

```
#include <stdio.h>
#define MAX 51
int main( ) {
        int matrix[MAX][MAX],i,j,k,lim,sum;
        do {   printf("\ndim. del quadrato ( dispari e <= %d ):“, MAX);
                scanf("%d",&lim);
        } while (( lim > MAX )||(lim % 2 == 0));
        for (i = 0; i < lim; i++) { for (j = 0; j < lim; j++) { matrix[i][j] = 0; } }
        i = lim – 1; j = lim / 2;
        for (k = 1; k <= lim*lim; k++) {
                matrix[i][j] = k;
                if (matrix[(i+1+lim)% lim][(j+1+lim) % lim] == 0) {
                        i = (i+1+lim) % lim;//il + lim serve a gestire
                                                //i numeri negativi
                        j = (j+1+lim) % lim;
                }  else
                        i = (i-1) % lim;
        }
```

```
        /*Visualizziamo quadrato e somma di ogni riga e colonna*/
        sum = 0;
        for (j = 0; j < lim; j++) {
                sum+=matrix[0][j];
        }
        printf("\nIl quadrato magico di ordine %d e':\n",lim);
        printf("\nLa somma su ogni linea e' uguale a %d.\n",sum);
        for (i = 0; i < lim; i++) {
                printf("\n");
                for (j = 0; j < lim; j++) {
                        printf("%4d",matrix[i][j]);
                }
        }
        return 0;
}
```

Prodotto di Kronecker

- Se A una matrice m×n e B una matrice p×q, allora il loro prodotto di Kronecker A x B è una matrice mxp×nxq definita a blocchi nel modo seguente:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

Cioè, esplicitando ogni termine:

$$A \otimes B = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix}$$

```c
#define N 3
#define M 4
#define P 5
#define Q 8
int main() {
  int a[N][M], b[P][Q], c[N*P][M*Q], i, j, k, t;
  //leggo la prima matrice
  for (i=0; i<N; i++)
    for (j=0; j<M; j++)
      scanf("%d", &a[i][j]);
  //leggo la seconda matrice
  for (i=0; i<P; i++)
    for (j=0; j<Q; j++)
      scanf("%d", &b[i][j]);
  //calcolo e il risultato
  for (i=0; i<N; i++)
    for (j=0; j<M; j++)
      for (k=0; k<P; j++)
        for (t=0; t<Q; t++)
          c[i*P+k][j*Q+t] = a[i][j]*b[k][t];
  //stampo
  for (i=0; i<N*P; i++){
    for (j=0; j<M*Q; j++){
      printf("%d ", c[i][j]);
    }
    printf("\n");
  }
  return 0;
}
```

Esercizio (tde 7-2-2012)

- Si scriva un programma che chiede all'utente di riempire una matrice NxN (con N costante globale predefinita) di interi e stampa la lunghezza della sequenza più lunga orizzontale, verticale o diagonale di numeri uguali consecutivi.

```
#define N 6
int main() {
  int m[N][N],i,j,cont=0,max=0;

  for(i=0;i<N;i++)
    for(j=1;j<N;j++)
        scanf("%d",&m[i][j]);

 //righe
  …
 //colonne
  …
  //diagonali
  …

  printf("%d",max+1);
}
```

```
//righe
for(i=0;i<N;i++){
   cont=0;
   for(j=1;j<N;j++)
      if(m[i][j]==m[i][j-1]){
        cont++;
        if(cont>max)
           max=cont;
     } else
       cont=0;
  }
```

```
//colonne
for(i=0;i<N;i++){
  cont=0;
  for(j=1;j<N;j++){
    if(m[j][i]==m[j-1][i]){
      cont++;
      if(cont>max)
        max=cont;
    } else {
      cont=0;
    }
  }
}
```

```
//diagonali

//diagonali da sx a dx parte sopra
for(i=0;i<N;i++){
  cont=0;
  for(j=1;j<N-i;j++){
    if(m[j][j+i]==m[j-1][j-1+i]){
      cont++;
      if(cont>max)
        max=cont;
    } else {
      cont=0;
    }
  }
}
```

```
//diagonali da sx a dx parte sotto
for(i=0;i<N;i++){
  cont=0;
  for(j=1;j<N-i;j++){
    if(m[j+i][j]==m[j-1+i][j-1]){
      cont++;
      if(cont>max)
        max=cont;
    } else {
      cont=0;
    }
  }
}
```

```
//diagonali da dx a sx parte sopra
for(i=0;i<N;i++){
  cont=0;
  for(j=1;j<N-i;j++){
    if(m[N-1-j][j+i]==m[N-1-(j-1)][j-1+i]){
      cont++;
      if(cont>max)
        max=cont;
    } else {
      cont=0;
    }
  }
}
```

```
//diagonali da dx a sx parte sotto
for(i=0;i<N;i++){
 cont=0;
 for(j=1;j<N-i;j++){
   if(m[j+i][N-1-j]==m[j-1+i][N-1-(j-1)]){
     cont++;
     if(cont>max)
       max=cont;
   } else {
     cont=0;
   }
  }
}
```

Esercizio (tde 23-2-2012)

- Si scriva un programma che chiede all'utente di riempire una matrice NxN (con N costante globale predefinita), un intero len (che deve essere un intero positivo maggiore di 1) e stampa OK se in m è presente almeno una sequenza orizzontale, verticale o diagonale, di lunghezza len, di elementi che crescono o diminuiscono linearmente (cioè in cui la differenza tra due elementi successivi è costante).
- Esempi di sequenze lineari:

  1 2 3 4 (lunghezza 4, differenza costante 1)
  4 3 2 1 (lunghezza 4, differenza constante -1)

  5 5 5 5 5 5 5 (lunghezza 7, differenza costante 0)
- Sono ammesse anche sequenze di lunghezza 1 (che è considerata sempre lineare)

Esempio (con matrice 5 per 5, per semplicità):

3	6	7	5	3
5	6	2	9	1
2	7	0	9	3
6	0	6	2	6
1	8	7	9	2

se len è 4, la funzione deve restituire 0, perché non c'è nessuna sequenza lineare di lunghezza 4, se len è 3, la funzione restituisce 1, perché è presente la sequenza orizzontale 7 5 3, con differenza costante -2

Esercizio

- Si realizzi un programma in linguaggio C che, data una matrice NxM di interi, trovi l'elemento per cui la media degli elementi ad esso adiacenti sia massima. Si stampino le coordinate di tale elemento ed il suo valore.
- Si considerino come adiacenti a ciascun elemento i quattro elementi nelle quattro direzioni cardinali. Si tratti inoltre l'ultima colonna come adiacente alla prima, e l'ultima riga come adiacente alla prima. Si supponga che N ed M possano variare tra 1 e 100. I valori di N ed M, così come i valori degli elementi della matrice, vengono acquisiti da tastiera.

Esercizio

- Scrivere un programma che chiede all'utente di inserire una matrice NxN e stampa gli elementi di tale matrice secondo un ordinamento a spirale, partendo dalla cornice più esterna e procedendo verso l'interno.

Esercizio (tde 5-2-2013)

- Si scriva una frammento di codice che usa una matrice di interi NxN (con N costante predefinita) e un array di caratteri.
- Ogni elemento dell'array contiene solo i caratteri '0', '1' o '\0' e rappresenta una stringa che è la codifica binaria di un intero. Il programma deve stampare VERO se il numero decimale corrispondente all'intero codificato in binario nell'array è uguale alla media degli interi contenuti nella matrice, FALSO altrimenti.