A.A. 08/09

Fondamenti di Programmazione

(canale E-O)

Docente: Prof.ssa Tiziana Calamoneri
calamo@di.uniroma1.it

Esercitatore: Dott. Roberto Petroccia
petroccia@di.uniroma1.it

Pagina del corso:
http://twiki.di.uniroma1.it/twiki/view/Programmazione1/EO/WebHome

---

Esercitazione del 19/11/08

Avviso

Gli studenti degli anni successivi al primo possono sostenere la prova di esonero su un solo canale

Lo stesso studente non può provare a sostenere due prove di esonero anche se poi ne consegna solo una

Indice

1. Soluzioni esercizi proposti mercoledì 12/11/08
2. Esercizi su ricorsione
3. Esercizi su matrici

19/11/08 - Roberto Petroccia: petroccia@di.uniroma1.it

Vettori bidimensionali

int v [5][7];

Chiediamo che venga allocata memoria per contenere 5*7 interi

v punta alla prima locazione di memoria allocata

ci muoviamo in memoria scorrendo gli indici di riga (primo indice) e di colonna (secondo indice)

Vettori bidimensionali

int v [5][5]

	0	1	2	3	4
0	v[0][0]	v[0][1]	v[0][2]	v[0][3]	v[0][4]
1	v[1][0]	v[1][1]	v[1][2]	v[1][3]	v[1][4]
2	v[2][0]	v[2][1]	v[2][2]	v[2][3]	v[2][4]
3	v[3][0]	v[3][1]	v[3][2]	v[3][3]	v[3][4]
4	v[4][0]	v[4][1]	v[4][2]	v[4][3]	v[4][4]

indice di colonna

indice di riga

nome vettore

Vettori bidimensionali

int v [5][5]

	0	1	2	3	4
0	v[0][0]	v[0][1]	v[0][2]	v[0][3]	v[0][4]
1	v[1][0]	v[1][1]	v[1][2]	v[1][3]	v[1][4]
2	v[2][0]	v[2][1]	v[2][2]	v[2][3]	v[2][4]
3	v[3][0]	v[3][1]	v[3][2]	v[3][3]	v[3][4]
4	v[4][0]	v[4][1]	v[4][2]	v[4][3]	v[4][4]

La memoria viene allocata in maniera sequenziale

v[0][0]	v[0][1]	v[0][2]	v[0][3]	v[0][4]	v[1][0]	v[1][1]	v[1][2]	v[1][3]	v[1][4]

Vettori bidimensionali

int v [5][5]

	0	1	2	3	4
0	v[0][0]	v[0][1]	v[0][2]	v[0][3]	v[0][4]
1	v[1][0]	v[1][1]	v[1][2]	v[1][3]	v[1][4]
2	v[2][0]	v[2][1]	v[2][2]	v[2][3]	v[2][4]
3	v[3][0]	v[3][1]	v[3][2]	v[3][3]	v[3][4]
4	v[4][0]	v[4][1]	v[4][2]	v[4][3]	v[4][4]

Qual è il secondo elemento della quarta riga?

Vettori bidimensionali

int v [5][5]

	0	1	2	3	4
0	v[0][0]	v[0][1]	v[0][2]	v[0][3]	v[0][4]
1	v[1][0]	v[1][1]	v[1][2]	v[1][3]	v[1][4]
2	v[2][0]	v[2][1]	v[2][2]	v[2][3]	v[2][4]
3	v[3][0]	v[3][1]	v[3][2]	v[3][3]	v[3][4]
4	v[4][0]	v[4][1]	v[4][2]	v[4][3]	v[4][4]

Qual è il secondo elemento della quarta riga?

v[3][1]

Vettori bidimensionali

int v [5][5]

	0	1	2	3	4
0	v[0][0]	v[0][1]	v[0][2]	v[0][3]	v[0][4]
1	v[1][0]	v[1][1]	v[1][2]	v[1][3]	v[1][4]
2	v[2][0]	v[2][1]	v[2][2]	v[2][3]	v[2][4]
3	v[3][0]	v[3][1]	v[3][2]	v[3][3]	v[3][4]
4	v[4][0]	v[4][1]	v[4][2]	v[4][3]	v[4][4]

Qual è il secondo elemento della quarta riga?

v[3][1]

Fate attenzione v[3][1] è diverso da v[1][3]

Vettori bidimensionali

Vettori bidimensionale come parametri di funzioni

int v [5][8];

Vettori bidimensionali

Vettori bidimensionale come parametri di funzioni

```
int v [5][8];
int funzione(int  v[ ][ ]) {
    ....................
}
```

non va bene, non so come muovermi in memoria

Vettori bidimensionali

Vettori bidimensionale come parametri di funzioni

```
int v [5][8];
int funzione(int  v[ ][ ]) {
    ....................
}
```

non va bene, non so come muovermi in memoria

La dimensione delle colonne deve sempre essere specificata

Vettori bidimensionali

Vettori bidimensionale come parametri di funzioni

```
int v [5][8];
int funzione(int  v[ ][ ]) {
    ....................
}
```

non va bene, non so come muovermi in memoria

La dimensione delle colonne deve sempre essere specificata

```
#define COL 10
int funzione(int  v[ ][COL ]) {
    ............
}
```

Vettori bidimensionali

Vettori bidimensionale come parametri di funzioni

```
int v [5][8];
int funzione(int  v[ ][ ]) {
    .....................
}
```

non va bene, non so come muovermi in memoria

La dimensione delle colonne deve sempre essere specificata

```
#define COL 10
int funzione(int  v[ ][COL ]) {
    .............
}
```

```
int funzione(int col, int  v[ ][col ]) {
    .............
}
```

Vettori bidimensionali

Vettori bidimensionale come parametri di funzioni

```
int v [5][8];
int funzione(int  v[ ][ ]) {
    .....................
}
```

non va bene, non so come muovermi in memoria

La dimensione delle colonne deve sempre essere specificata

```
int funzione(int row, int col, int  v[ ][col ]) {
    .............
}

int funzione(int row, int col, int  v[row ][col ]) {
    .............
}
```

Esercizi

1. Scrivere una funzione che preso un array ed una dimensione (letta da input) riempia l'array usando la funzione rand().

2. Scrivere una funzione che stampi un array di interi.

3. Scrivere una funzione che presa una matrice ed una dimensione per riga e colonna (lette da input) riempia la matrice usando la funzione rand().

4. Scrivere una funzione che stampi una matrice di interi.

5. Scrivere una funzione che stampi la somma degli elementi di una matrice di interi.

6. Scrivere una funzione che stampi solo gli elementi che si trovano in una riga e colonna pari di una matrice.

7. Implementare il selection sort.

8. Implementare la ricerca binaria.

1. Scrivere una funzione che preso un array ed una dimensione (letta da input) riempia l'array usando la funzione rand().

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VALUE 100
void riempiArray (int V[], int n) {
      int i;
      for (i = 0; i < n; i++) {
         V[i] = rand() % MAX_VALUE;
      }
   }

int main() {
        int a;
        scanf (“%d”, &a);
        int b[a];
        riempiArray(b, a);
        return 0;
}
```

1. Scrivere una funzione che preso un array ed una dimensione (letta da input) riempia l'array usando la funzione rand().

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VALUE 100
void riempiArray (int V[], int n) {
      int i;
      for (i = 0; i < n; i++) {
         V[i] = rand() % MAX_VALUE;
      }
   }

int main() {
      int a;
      scanf (“%d”, &a);
      int b[a];
      riempiArray(b, a);
      return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VALUE 100
void riempiArray2 (int V[], int n, int i) {
   if (i >= n) {
      return;
   }
   V[i] = rand() % MAX_VALUE;
      riempiArray2(V, n, i+1);
}
int main() {
   int a;
   scanf (“%d”, &a);
   int b[a];
   riempiArray2(b, a, 0);
   return 0;
}
```

2. Scrivere una funzione che stampi un array di interi.

```
#include <stdio.h>
void stampaArray (int V[], int n) {
      int i;
      for (i = 0; i < n; i++) {
         printf(“%d ”, V[i]);
      }
      printf(“\n”);
   }

int main() {
        int a;
        scanf (“%d”, &a);
        int b[a];
        stampaArray(b, a);
        return 0;
}
```

2. Scrivere una funzione che stampi un array di interi.

```
#include <stdio.h>
void stampaArray (int V[], int n) {
      int i;
      for (i = 0; i < n; i++) {
         printf(“%d ”, V[i]);
      }
      printf(“\n”);
   }

int main() {
         int a;
         scanf (“%d”, &a);
         int b[a];
         stampaArray(b, a);
         return 0;
}
```

```
#include <stdio.h>
void stampaArray2(int V[], int n, int i) {
   if (i >= n) {
      printf(“\n”);
      return;
   }
   printf(“%d ”, V[i]);
      stampaArray2(V, n, i+1);
}
int main() {
   int a;
   scanf (“%d”, &a);
   int b[a];
   stampaArray2(b, a, 0);
   return 0;
}
```

3. Scrivere una funzione che presa una matrice ed una dimensione per riga e colonna (lette da input) riempia la matrice usando la funzione rand().

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VALUE 100
void riempiMatrice (int r, int c, int M[][c]) {
    int i, j;
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            M[i][j] = rand() % MAX_VALUE;
        }
    }
}
int main() {
        int x, y;
        scanf (“%d%d”, &x,&y);
        int M[x][y];
        riempiMatrice(x, y, M);
        return 0;
}
```

3. Scrivere una funzione che presa una matrice ed una dimensione per riga e colonna (lette da input) riempia la matrice usando la funzione rand().

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VALUE 100
void riempiMatrice (int r, int c, int M[][c]) {
    int i, j;
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            M[i][j] = rand() % MAX_VALUE;
        }
    }
}
int main() {
        int x, y;
        scanf (“%d%d”, &x,&y);
        int M[x][y];
        riempiMatrice(x, y, M);
        return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VALUE 100
void riempiMatrice2 (int r, int c,
                     int M[][c], int i, int j) {
    if (i >= r) {
        return;
    }
    if (j >= c) {
        riempiMatrice2(r, c, M, i+1, 0);
        return;
    }
    M[i][j] = rand() % MAX_VALUE;
    riempiMatrice2(r, c, M, i, j+1);
}
int main() {
        int x, y;
        scanf (“%d%d”, &x,&y);
        int M[x][y];
        riempiMatrice2(x, y,M, 0, 0);
        return 0;
}
```

4. Scrivere una funzione che stampi un matrice di interi.

```
#include <stdio.h>
void stampaMatrice (int r, int c, int M[][c]) {
    int i, j;
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            printf("%3.d ", M[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}
int main() {
        int x, y;
        scanf ("%d%d", &x,&y);
        int M[x][y];
        riempiMatrice(x, y, M);
        stampaMatrice(x, y, M);
        return 0;
}
```

4. Scrivere una funzione che stampi un matrice di interi.

```
#include <stdio.h>
void stampaMatrice (int r, int c, int M[][c]) {
    int i, j;
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            printf("%3.d ", M[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}
int main() {
        int x, y;
        scanf ("%d%d", &x,&y);
        int M[x][y];
        riempiMatrice(x, y, M);
        stampaMatrice(x, y, M);
        return 0;
}
```

```
#include <stdio.h>
void stampaMatrice2 (int r, int c,
                     int M[][c], int i, int j) {
    if (i >= r) {
        printf("\n");
        return;
    }
    if (j >= c) {
        printf("\n");
        stampaMatrice2(r, c, M, i+1, 0);
        return;
    }
    printf("%3.d ", M[i][j]);
    stampaMatrice2(r, c, M, i, j+1);
}
int main() {
        int x, y;
        scanf ("%d%d", &x,&y);
        int M[x][y];
        riempiMatrice2(x, y,M, 0, 0);
        stampaMatrice2(x, y,M, 0, 0);
        return 0;
}
```

5. Scrivere una funzione che stampi la somma degli elementi di una matrice di interi.

```
#include <stdio.h>
int sommaMatrice (int r, int c, int M[][c]) {
    int i, j;
    int somma = 0;
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            somma += M[i][j]);
        }
    }
    return somma;
}
int main() {
        int x, y;
        scanf (“%d%d”, &x,&y);
        int M[x][y];
        riempiMatrice(x, y, M);
        stampaMatrice(x, y, M);
        printf(“%d\n”, sommaMatrice(x,y,M));
        return 0;
}
```

5. Scrivere una funzione che stampi la somma degli elementi di una matrice di interi.

```
#include <stdio.h>
int sommaMatrice (int r, int c, int M[][c]) {
    int i, j;
    int somma = 0;
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            somma += M[i][j]);
        }
    }
    return somma;
}
int main() {
        int x, y;
        scanf (“%d%d”, &x,&y);
        int M[x][y];
        riempiMatrice(x, y, M);
        stampaMatrice(x, y, M);
        printf(“%d\n”, sommaMatrice(x,y,M));
        return 0;
}
```

```
#include <stdio.h>
int sommaMatrice2 (int r, int c,
                   int M[][c], int i, int j) {
    if (i >= r) {
        return 0;
    }
    if (j >= c) {
        return sommaMatrice2(r, c, M, i+1, 0);
    }
    return M[i][j] + sommaMatrice2(r, c, M, i, j+1);
}

int main() {
        int x, y;
        scanf (“%d%d”, &x,&y);
        int M[x][y];
        riempiMatrice2(x, y,M, 0, 0);
        stampaMatrice2(x, y,M, 0, 0);
        printf(“%d\n”, sommaMatrice(x,y,M, 0, 0));
        return 0;
}
```

7.Implementare il selection sort.

PSEUDOCODICE
Selection(Vett, dim)
per ogni i da 0 a n-2:
    trova il minimo nel sottovettore [i, n-1]
    scambia il minimo trovato con Vett[i]

7.Implementare il selection sort.

```
#include <stdio.h>
void selectionSort (int A[], int n) {
    int i, j;
    int aux;
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (A[i] > A[j]) {
                aux = A[i];
                A[i] = A[j];
                A[j] = aux;
            }
        }
    }
}
int main() {
        int x;
        scanf (“%d”, &x);
        int V[x];
        riempiVettore(V, x);
        selectionSort(V, x);
        return 0;
}
```

7.Implementare il selection sort.

```
#include <stdio.h>
void selectionSort (int A[], int n) {
    int i, j;
    int aux;
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (A[i] > A[j]) {
                aux = A[i];
                A[i] = A[j];
                A[j] = aux;
            }
        }
    }
}
int main() {
    int x;
    scanf (“%d”, &x);
    int V[x];
    riempiVettore(V, x);
    selectionSort(V, x);
    return 0;
}
```

```
#include <stdio.h>
void selectionSort2 (int A[], int n, int i) {
    int j;
    int aux;
    if (i >= n) {
        return;
    }
    for (j = i + 1; j < n; j++) {
        if (A[i] > A[j]) {
            aux = A[i];
            A[i] = A[j];
            A[j] = aux;
        }
    }
    selectionSort2(A, n, i+1);
}
int main() {
    int x;
    scanf (“%d”, &x);
    int V[x];
    riempiVettore(V, x);
    selectionSort2(V, x, 0);
    return 0;
}
```

8.Implementare la ricerca binaria.

```
#include <stdio.h>
int binarySearch (int A[], int dim, int k) {
    int min_index = 0;
    int max_index = dim - 1;
    while (min_index <= max_index) {
        int middle = (max_index + min_index)/2;
        if (k < A[middle])
            max_index = middle - 1;
        else if (k > A[middle])
            min_index = middle + 1;
        else return 1;
    }
    return 0;
}
```

```
int main() {
        int x, k;
        scanf (“%d%d”, &x, &k);
        int V[x];
        riempiVettore(V, x);
        selectionSort(V, x);
        binarySearch(V, x, k)
        return 0;
    }
```

8.Implementare la ricerca binaria.

```
#include <stdio.h>
int binarySearch2 (int A[], int dim, int k,
                        int min_index, int max_index) {
    if (min_index >= max_index) {
        return 0;
    }
    int middle = (max_index + min_index)/2;
    if (k < A[middle]) {
        return binarySearch2(A, dim, k, min_index, middle - 1);
    }
    else if (k > A[middle]) {
        return binarySearch2(A, dim, k, middle + 1, max_index);
    }
    else return 1;
}

  int main() {
        int x, k;
        scanf (“%d%d”, &x, &k);
        int V[x];
        riempiVettore(V, x);
        selectionSort(V, x);
        binarySearch2(V, x, k, 0, x-1)
        return 0;
  }
```

DOMANDE ???

19/11/08 - Roberto Petroccia: petroccia@di.uniroma1.it

Esercizi ricorsione

1. Scrivere una funzione che calcoli la somma degli elementi di un array in maniera ricorsiva.

2. Scrivere una funzione che conti il numero delle occorrenze di un valore k in un vettore in maniera ricorsiva.

3. Scrivere una funzioni che preso un vettore di interi lo inverta. Implementare la versione iterativa e quella ricorsiva.

4. Scrivere una funzione merge che presi in input due array ordinati li fonda in un unico array ordinato. Tale funzione avrà tra i parametri tre array, due sono quelli da fondere ed il terzo conterrà il risultato. Implementare la versione iterativa e ricorsiva.

5. Scrivere una funzione che controlli se un vettore è palindromo oppure no. Un vettore si dice palindromo se leggendolo da sinistra a destra o da destra a sinistra si ha la stessa cosa. Ad esempio 1 2 2 1 oppure 1 2 1 sono palindromi, 1 2 3 4 non è palindromo. Scrivere la versione iterativa e ricorsiva

19/11/08 - Roberto Petroccia: petroccia@di.uniroma1.it

Esercizi matrici

1. Scrivere una funzione che presa in input una matrice ed un valore k restituisca il numero di volte che k compare nella matrice.

2. Scrivere una funzione che presa in input una matrice quandrata dica se tale matrice rappresenta un quadrato magico o meno. Un quadrato magico di dimensione N è una matrice in cui sono presenti tutti e soli i numeri tra 1 e N*N, disposti in modo tale che la somma dei valori su tutte le righe è costante, ed è pari alla somma dei valori su tutte le colonne.

3. Scrivere una funzione che presa in input una matrice, le sue dimensioni e due puntatori a intero memorizzi in tali puntatori il valore dell'indice di riga e colonna dell'elemento più grande della matrice.

19/11/08 - Roberto Petroccia: petroccia@di.uniroma1.it

1. Scrivere una funzione che calcoli la somma degli elementi di un array in maniera ricorsiva.

```
#include <stdio.h>
int sommaArray (int V[], int n) {
      int i;
      int somma = 0;
      for (i = 0; i < n; i++) {
         somma += V[i];
      }
      return somma;
   }

int main() {
        int a;
        scanf (“%d”, &a);
        int b[a];
        int p = sommaArray(b, a);
        return 0;
}
```

```
#include <stdio.h>
int sommaArray2 (int V[], int n, int i) {
    if ( i >= n)
        return 0;

    return V[i] + sommaArray2(V, n, i+1);
}

int main() {
    int a;
    scanf (“%d”, &a);
    int b[a];
    int p = sommaArray2(b, a, 0);
    return 0;
}
```

2. Scrivere una funzione che conti il numero delle occorrenze di un valore k in un vettore in maniera ricorsiva.

```
#include <stdio.h>
int countArray (int V[], int n, int k) {
      int i;
      int count = 0;
      for (i = 0; i < n; i++) {
         if (V[i] == k)
            count ++;
      }
      return count;
   }

int main() {
        int a;
        scanf (“%d”, &a);
        int b[a];
        int p = countArray(b, a, 3);
        return 0;
}
```

```
#include <stdio.h>
int countArray2 (int V[], int n, int i, int k) {
    if ( i >= n)
        return 0;
    if (V[i] == k)
        return 1 + countArray2(V, n, i+1, k);
    return countArray2(V, n, i+1, k);
}

int main() {
    int a;
    scanf (“%d”, &a);
    int b[a];
    int p = countArray2(b, a, 0, 3);
    return 0;
}
```

3. Scrivere una funzioni che preso un vettore di interi lo inverta. Implementare la versione iterativa e quella ricorsiva.

```
#include <stdio.h>
void invertiArray (int V[], int n) {
      int i, aux;
      for (i = 0; i < n/2; i++) {
         aux = V[i];
         V[i] = V[n - 1 - i];
         V[n - 1 -i] = aux;
      }
}

int main() {
         int a;
         scanf (“%d”, &a);
         int b[a];
         invertiArray(b, a);
         return 0;
}
```

```
#include <stdio.h>
void invertiArray2 (int V[], int n) {
   if (n <= 0){
      return;
   }
   int aux;
   aux = V[0];
   V[0] = V[n - 1];
   V[n - 1] = aux;
   invertyArray2(++V, n-2);
}

int main() {
   int a;
   scanf (“%d”, &a);
   int b[a];
   invertiArray2(b, a);
   return 0;
}
```

4. Funzione merge (iterativa)

```
void mergeArray2(int A[], int dim_a, int B[], int dim_b, int C[]){
    if (dim_a <= 0 && dim_b <= 0) {
        return;
    }
    if (dim_a <= 0) {
        *C = *B;
        mergeArray2(A, dim_a, ++B, dim_b - 1, ++C);
        return;
    }
    if (dim_b <= 0) {
        *C = *A;
        mergeArray2(++A, dim_a - 1, B, dim_b, ++C);
        return;

    }
    if (*A <= *B) {
        *C = *A;
        mergeArray2(++A, dim_a - 1, B, dim_b, ++C);
    }
    else {
        *C = *B;
        mergeArray2(A, dim_a, ++B, dim_b - 1, ++C);
    }
}
```

```
int main() {
    int x, y;
    scanf (“%d%d”, &x,&y);
    int A[x];
    int B[y];
    int C[x + y];
    mergeArray2(A,x,B,y,C);
    return 0;
}
```

4. Funzione merge (ricorsiva)

```
void mergeArray(int A[], int dim_a, int B[], int dim_b, int C[]){
    int i = 0;
    int j = 0;
    int k = 0;
    while (i < dim_a && j < dim_b) {
        if (A[i] <= B[j]) {
            C[k] = A[i];
            i++;
        }
        else {
            C[k] = B[j];
            j++;
        }
        k++;
    }
    while (i < dim_a) {
        C[k] = A[i];
        i++;
        k++;
    }
    while (j < dim_b) {
        C[k] = B[j];
        j++;
        k++;
    }
}
```

```
int main() {
        int x, y;
        scanf ("%d%d", &x,&y);
        int A[x];
        int B[y];
        int C[x + y];
        mergeArray(A,x,B,y,C);
        return 0;
}
```

5. Scrivere una funzione che controlli se un vettore è palindromo oppure no.

```
#include <stdio.h>
int palindromo (int V[], int n) {
        int i;
        for (i = 0; i < n/2; i++) {
            if (V[i] != V[n - 1 - i])
                return 0;
        }
        return 1;
}

int main() {
        int a;
        scanf (“%d”, &a);
        int b[a];
        int p = palindromo(b, a);
        return 0;
}
```

```
#include <stdio.h>
int palindromo2 (int V[], int n) {
    if (n <= 0){
        return 1;
    }
    if (V[0] == V[n - 1]) {
        return palindromo2(++V, n-2);
    }
    else
        return 0;
}

int main() {
    int a;
    scanf (“%d”, &a);
    int b[a];
    int p = palindromo2(b, a);
    return 0;
}
```

1. Scrivere una funzione che presa in input una matrice ed un valore k restituisca il numero di volte che k compare nella matrice.

```
#include <stdio.h>
int countMatrice (int r, int c, int M[][c], int k) {
        int i, j;
        int count = 0;
        for (i = 0; i < r; i++) {
            for (j = 0; j < c; j++) {
                if (M[i][j] == k)
                    count ++;
            }
        }
        return count;
}
int main() {
        int x, y, k;
        scanf (“%d%d%d”, &x, &y, &k);
        int M[x][y];
        riempiMatrice(x, y, M);
        int p = countMatrice(x, y, M, k);
        return 0
}
```

2. Quadrato magico

```
#include <stdio.h>
int magicMatrix (int n, int M[][n]) {
        int i, j;
        int somma = 0;
        int somma_aux;
        int A[n*n];
        for (i = 0; i < n*n; i++) {
            A[i] = 0;
        }
        for (i = 0; i < r; i++) {
            somma_aux = 0;
        for (j = 0; j < c; j++) {
            if (i == 0) {
                somma += M[i][j];
            }
            somma_aux += M[i][j];
            if (A[(i*j)+j] != 0) {
                return 0;
            }
            A[(i*j)+j] = 1;
        }
        if (somma != somma_aux) {
            return 0;
        }
}
```

```
        for (j = 0; j < c; j++) {
            somma_aux = 0;
            for (i = 0; i < r; i++) {
                somma_aux += M[i][j];
            }
            if (somma != somma_aux) {
                return 0;
            }
        }
        return 1;
}
```

3. Scrivere una funzione che presa in input una matrice, le sue dimensioni e due puntatori a intero memorizzi in tali puntatori il valore dell'indice di riga e colonna dell'elemento più grande della matrice.

```
#include <stdio.h>
void minMatrice (int r, int c, int M[][c], int* min_r, int* min_c) {
        int i, j, min;
        min = M[0][0];
        *min_r = 0;
        *min_c = 0;
        for (i = 0; i < r; i++) {
            for (j = 0; j < c; j++) {
                if (M[i][j] < min)
                    min = M[i][j];
                    *min_r = i;
                    *min_c = j;
            }
        }
}
int main() {
        int x, y, min_r, min_c;
        scanf ("%d%d", &x, &y);
        int M[x][y];
        riempiMatrice(x, y, M);
        minMatrice(x, y, M, &min_r, &min_c);
        return 0;
}
```