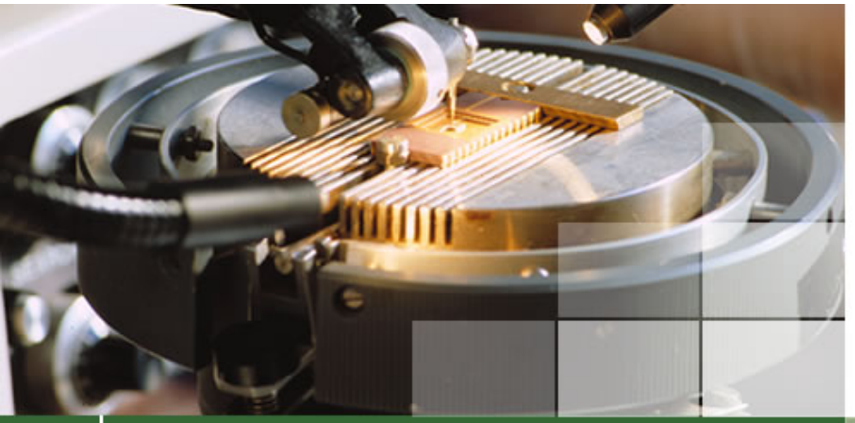


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Array e matrici

Matteo Ferroni

matteo.ferroni@polimi.it

26/10/2018

Agenda

(75’) Array n-dimensionali: le matrici

(15’) Pausa

(40’) Matrici - Esercizi vari

(20’) Metodo di Eratostene *(scorsa esercitazione)*

Esercizio: Matrici

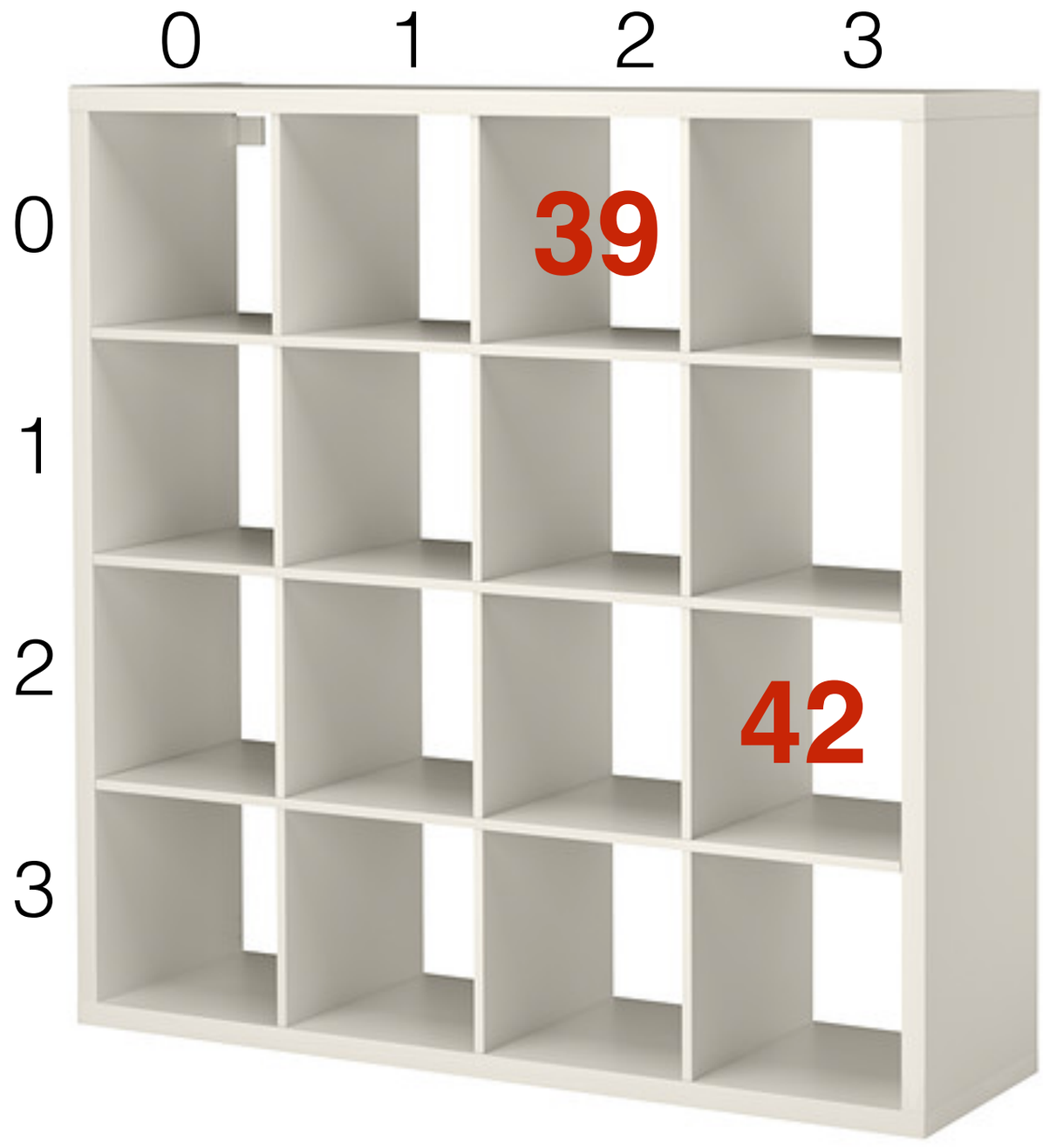
- Scrivere un programma che legge una matrice quadrata di dimensioni specificate dall'utente (al massimo 10 righe e 10 colonne):
  - Calcolare la somma dei valori sulle **righe**
  - Calcolare la somma dei valori sulle **colonne**
  - calcolare la somma dei valori sulla **diagonale** principale
  - calcolare la somma dei valori **sopra la diagonale** principale
  - calcolare la somma dei valori **sotto la diagonale** principale

Matrici nel mondo reale

```
int matrice[4][4];

matrice[0][2] = 39;

    int i = 2;
    int j = 3;
matrice[i][j] = 42;
```



Esercizio: Matrici

- Scrivere un programma che legge una matrice quadrata di dimensioni specificate dall'utente (al massimo 10 righe e 10 colonne):
  - Calcolare la somma dei valori sulle **righe**
  - Calcolare la somma dei valori sulle **colonne**
  - calcolare la somma dei valori sulla **diagonale** principale
  - calcolare la somma dei valori **sopra la diagonale** principale
  - calcolare la somma dei valori **sotto la diagonale** principale

Esercizio: Matrici

Esercizio: Matrici

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

#define MAXDIM 10
```

Esercizio: Matrici

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

#define MAXDIM 10

int main(){
    int matrice[MAXDIM][MAXDIM];
```

Esercizio: Matrici

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

#define MAXDIM 10

int main(){
    int matrice[MAXDIM][MAXDIM];
    int righe[MAXDIM];
    int colonne[MAXDIM];
```

Esercizio: Matrici

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

#define MAXDIM 10

int main(){
    int matrice[MAXDIM][MAXDIM];
    int righe[MAXDIM];
    int colonne[MAXDIM];
    int sup = 0, inf = 0, diag = 0, dim = 0, i = 0, j = 0;
```

Esercizio: Matrici

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

#define MAXDIM 10

int main(){
    int matrice[MAXDIM][MAXDIM];
    int righe[MAXDIM];
    int colonne[MAXDIM];
    int sup = 0, inf = 0, diag = 0, dim = 0, i = 0, j = 0;
```

```c
    printf("inserire il numero di righe della matrice (massimo 10)\n");
    scanf("%d", &dim);
```

Esercizio: Matrici

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

#define MAXDIM 10

int main(){
    int matrice[MAXDIM][MAXDIM];
    int righe[MAXDIM];
    int colonne[MAXDIM];
    int sup = 0, inf = 0, diag = 0, dim = 0, i = 0, j = 0;
```

```c
do{
    printf("inserire il numero di righe della matrice (massimo 10)\n");
    scanf("%d", &dim);
    }while(dim < 1 || dim > 10);
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
```

```
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
```

```
        }

}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);
    }
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);
    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);

        righe[i] = righe[i] + matrice[i][j];
    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);

        righe[i] = righe[i] + matrice[i][j];

        colonne[j] = colonne[j] + matrice[i][j];

    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);

        righe[i] = righe[i] + matrice[i][j];

        colonne[j] = colonne[j] + matrice[i][j];

        if(i==j)

    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);

        righe[i] = righe[i] + matrice[i][j];

        colonne[j] = colonne[j] + matrice[i][j];

        if(i==j)
            diag = diag + matrice[i][j];

    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);

        righe[i] = righe[i] + matrice[i][j];

        colonne[j] = colonne[j] + matrice[i][j];

        if(i==j)
            diag = diag + matrice[i][j];
        else if (i < j)

    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);

        righe[i] = righe[i] + matrice[i][j];

        colonne[j] = colonne[j] + matrice[i][j];

        if(i==j)
            diag = diag + matrice[i][j];
        else if (i < j)
            sup = sup + matrice[i][j];

    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);

        righe[i] = righe[i] + matrice[i][j];

        colonne[j] = colonne[j] + matrice[i][j];

        if(i==j)
            diag = diag + matrice[i][j];
        else if (i < j)
            sup = sup + matrice[i][j];
        else
            inf = inf + matrice[i][j];
    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare la matrice a video e calcolare le somme di righe, colonne,
    diagonale e triangolari superiori ed inferiori **/
for(i=0; i<dim; i++){
    for(j=0; j<dim; j++){
        printf("%d ", matrice[i][j]);

        righe[i] = righe[i] + matrice[i][j];

        colonne[j] = colonne[j] + matrice[i][j];

        if(i==j)
            diag = diag + matrice[i][j];
        else if (i < j)
            sup = sup + matrice[i][j];
        else
            inf = inf + matrice[i][j];
    }
    printf("\n");
}
```

Esercizio: Matrici

```
/** stampare le somme delle righe**/
```

Esercizio: Matrici

```
/** stampare le somme delle righe**/
for(i=0; i<dim; i++){
```

Esercizio: Matrici

```
/** stampare le somme delle righe**/
for(i=0; i<dim; i++){
    printf("somma riga %d = %d\n", i+1, righe[i]);
}
```

Esercizio: Matrici

```
/** stampare le somme delle righe**/
for(i=0; i<dim; i++){
    printf("somma riga %d = %d\n", i+1, righe[i]);
}

/** stampare le somme delle colonne**/
```

Esercizio: Matrici

```
/** stampare le somme delle righe**/
for(i=0; i<dim; i++){
    printf("somma riga %d = %d\n", i+1, righe[i]);
}

/** stampare le somme delle colonne**/
for(j=0; j<dim; j++){
```

Esercizio: Matrici

```c
/** stampare le somme delle righe**/
for(i=0; i<dim; i++){
    printf("somma riga %d = %d\n", i+1, righe[i]);
}

/** stampare le somme delle colonne**/
for(j=0; j<dim; j++){
    printf("somma colonna %d = %d\n", j+1, colonne[j]);
}
```

Esercizio: Matrici

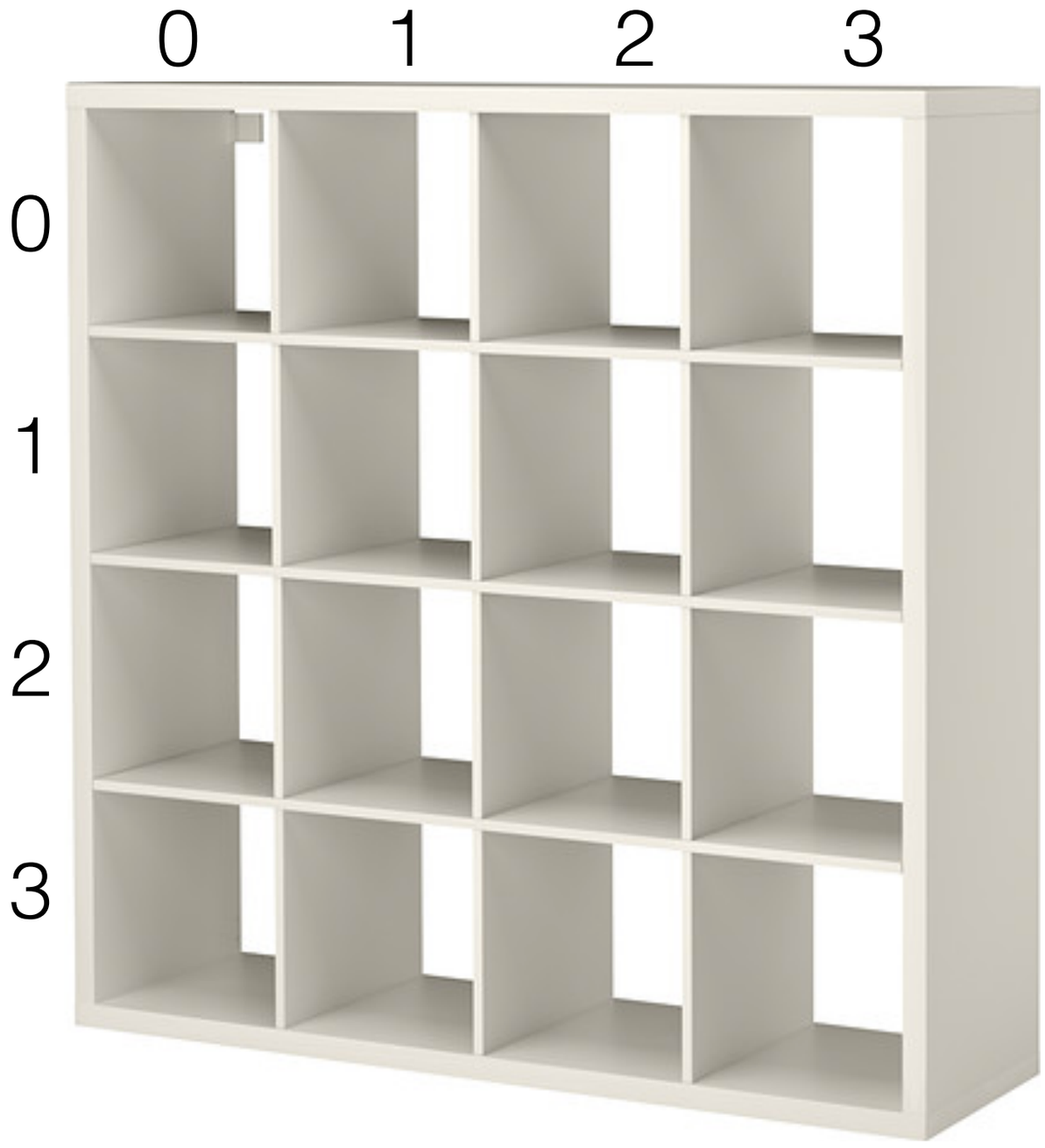
```c
/** stampare le somme delle righe**/
for(i=0; i<dim; i++){
    printf("somma riga %d = %d\n", i+1, righe[i]);
}

/** stampare le somme delle colonne**/
for(j=0; j<dim; j++){
    printf("somma colonna %d = %d\n", j+1, colonne[j]);
}

printf("la somma delle diagonale e': %d\n", diag);
printf("la somma della parte superiore alla diagonale e': %d\n", sup);
printf("la somma della parte inferiore alla diagonale e': %d\n", inf);
```

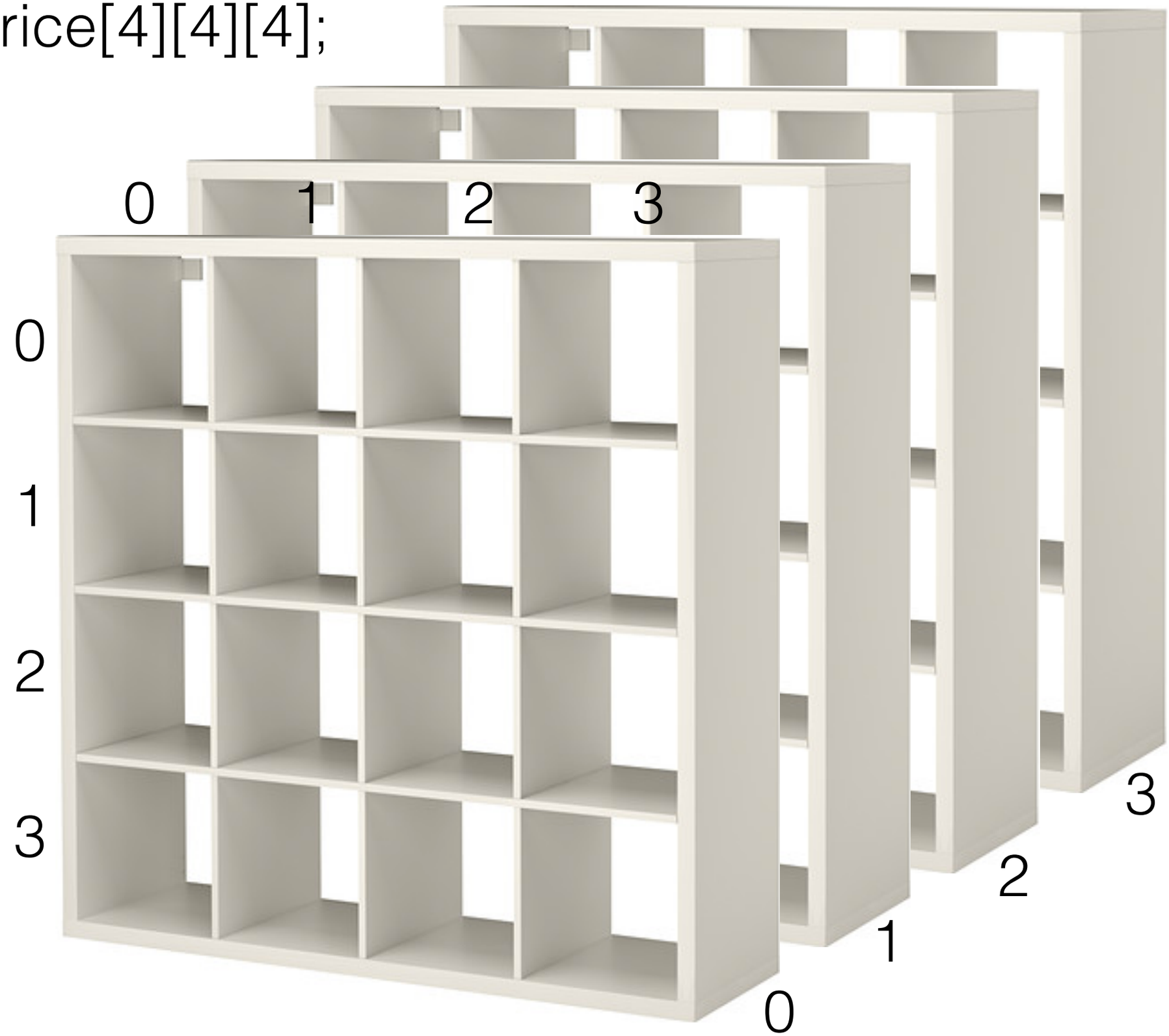
Matrici nel mondo reale

int matrice[4][4];



Matrici nel mondo reale

int matrice[4][4][4];



Esercizio: Matrici multidimensionali

Sia dato un array multidimensionale char a[1000][34][131]. Contare tutte le occorrenze delle sole vocali e stamparne a schermo la quantità, per ogni vocale.

```
#define X 1000
#define Y 34
#define Z 131

char a[X][Y][Z];
int vocali[5]={0};
int x,y,z;
```

Esercizio: Matrici multidimensionali

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
```

```
}
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
```

```
        }
    }
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
        for(z=0 ; z<Z; z++){
```

```
        }
    }
}
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
        for(z=0 ; z<Z; z++){

            if(a[x][y][z]=='a'|| a[x][y][z]=='e'||
               a[x][y][z]=='i'|| a[x][y][z]=='o'||
               a[x][y][z]=='u' ){
                    }
                }
        }
}
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
        for(z=0 ; z<Z; z++){

            if(a[x][y][z]=='a'|| a[x][y][z]=='e'||
               a[x][y][z]=='i'|| a[x][y][z]=='o'||
               a[x][y][z]=='u' ){

                switch(a[x][y][z]){
```

```
                }
            }
        }
    }
}
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
        for(z=0 ; z<Z; z++){

            if(a[x][y][z]=='a'|| a[x][y][z]=='e'||
               a[x][y][z]=='i'|| a[x][y][z]=='o'||
               a[x][y][z]=='u' ){

                switch(a[x][y][z]){
                    case 'a': vocali[0]++; break;
```

```
                }
            }
        }
    }
}
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
        for(z=0 ; z<Z; z++){

            if(a[x][y][z]=='a'|| a[x][y][z]=='e'||
               a[x][y][z]=='i'|| a[x][y][z]=='o'||
               a[x][y][z]=='u' ){

                switch(a[x][y][z]){
                    case 'a': vocali[0]++; break;
                    case 'e': vocali[1]++; break;
```

```
                }
            }
        }
    }
}
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
        for(z=0 ; z<Z; z++){

            if(a[x][y][z]=='a'|| a[x][y][z]=='e'||
               a[x][y][z]=='i'|| a[x][y][z]=='o'||
               a[x][y][z]=='u' ){

                switch(a[x][y][z]){
                    case 'a': vocali[0]++; break;
                    case 'e': vocali[1]++; break;
                    case 'i': vocali[2]++; break;
```

```
                }
            }
        }
    }
}
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
        for(z=0 ; z<Z; z++){

            if(a[x][y][z]=='a'|| a[x][y][z]=='e'||
               a[x][y][z]=='i'|| a[x][y][z]=='o'||
               a[x][y][z]=='u' ){

                switch(a[x][y][z]){
                    case 'a': vocali[0]++; break;
                    case 'e': vocali[1]++; break;
                    case 'i': vocali[2]++; break;
                    case 'o': vocali[3]++; break;
```

```
                }
            }
        }
    }
}
```

Esercizio: Matrici multidimensionali

```
for(x=0; x<X; x++){
    for(y=0; y<Y; y++){
        for(z=0 ; z<Z; z++){

            if(a[x][y][z]=='a'|| a[x][y][z]=='e'||
               a[x][y][z]=='i'|| a[x][y][z]=='o'||
               a[x][y][z]=='u' ){

                switch(a[x][y][z]){
                    case 'a': vocali[0]++; break;
                    case 'e': vocali[1]++; break;
                    case 'i': vocali[2]++; break;
                    case 'o': vocali[3]++; break;
                    case 'u': vocali[4]++; break;
                }
            }
        }
    }
}
```

Esercizio: Matrici multidimensionali

```
printf("\n\nLe a sono: %d", vocali[0]);
printf("\nLe e sono: %d", vocali[1]);
printf("\nLe i sono: %d", vocali[2]);
printf("\nLe o sono: %d", vocali[3]);
printf("\nLe u sono: %d\n\n", vocali[4]);
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

Scrivere un programma che stampa i numeri primi minori di 100.
(Crivello di Eratostene)

Metodo di Eratostene

1. Si scrivono tutti i numeri naturali a partire da 2 fino ad N in un elenco detto “setaccio”.
2. Poi si cancellano (setacciano) tutti i multipli del primo numero del setaccio (escluso lui stesso).
3. Si prosegue così fino ad arrivare in fondo.
4. I numeri che restano sono i numeri primi minori od uguali a n.

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;
```

```c
    for (i=1;i<100;i++){
        if(array[i]==1)
            printf("\t%d", i);
    }
    return 0;
}
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;

    // Si procede al setaccio: ignoro i primi due elementi (numeri 0 e 1)
    for (i=2;i<100;i++) {
```

```c
    for (i=1;i<100;i++){
        if(array[i]==1)
            printf("\t%d", i);
    }
    return 0;
}
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;

    // Si procede al setaccio: ignoro i primi due elementi (numeri 0 e 1)
    for (i=2;i<100;i++) {
        // Se il numero è ancora marcato come primo (non è multiplo dei suoi precedenti)
        if(array[i]==1) {
```

```c
    for (i=1;i<100;i++){
        if(array[i]==1)
            printf("\t%d", i);
    }
    return 0;
}
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;

    // Si procede al setaccio: ignoro i primi due elementi (numeri 0 e 1)
    for (i=2;i<100;i++) {
        // Se il numero è ancora marcato come primo (non è multiplo dei suoi precedenti)
        if(array[i]==1) {
            // Marco come "non primi" tutti i suoi multipli
```

```c
    for (i=1;i<100;i++){
        if(array[i]==1)
            printf("\t%d", i);
    }
    return 0;
}
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;

    // Si procede al setaccio: ignoro i primi due elementi (numeri 0 e 1)
    for (i=2;i<100;i++) {
        // Se il numero è ancora marcato come primo (non è multiplo dei suoi precedenti)
        if(array[i]==1) {
            // Marco come "non primi" tutti i suoi multipli
            for (j=2; j<=(100/i); j++){
                array[i*j] = 0;
            }
        }
    }

    // Si stampano i primi 100 numeri primi
    printf("\nNumeri primi < 100: ");

    for (i=1;i<100;i++){
        if(array[i]==1)
            printf("\t%d", i);
    }
    return 0;
}
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;

    // Si procede al setaccio: ignoro i primi due elementi (numeri 0 e 1)
    for (i=2;i<100;i++) {
        // Se il numero è ancora marcato come primo (non è multiplo dei suoi precedenti)
        if(array[i]==1) {
            // Marco come "non primi" tutti i suoi multipli
            for (j=2; j<=(100/i); j++){
                array[i*j] = 0;
            }
        }
    }

    // Si stampano i primi 100 numeri primi
    printf("\nNumeri primi < 100: ");

    for (i=1;i<100;i++){
        if(array[i]==1)
            printf("\t%d", i);
    }
    return 0;
}
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;

    // Se il numero è ancora marcato come primo (non è multiplo dei suoi precedenti)
    if(array[i]==1) {
        // Marco come "non primi" tutti i suoi multipli
        for (j=2; j<=(100/i); j++){
            array[i*j] = 0;
        }
    }

    printf( "\nNumeri primi < 100: " );

    for (i=1;i<100;i++){
        if(array[i]==1)
            printf("\t%d", i);
    }
    return 0;
}
```

Esercizio 4: Metodo di Eratostene

SCORSA ESERCITAZIONE

```c
#include <stdio.h>
int main()
{
    int i,j;

    // Utilizzo un array di 100 elementi
    // Se il contenuto nella cella i-esima è 1, il numero è primo
    int array[100];

    // Inizializzazione dell'array: "tutti i numeri sono primi"
    for (i=0;i<100;i++)
        array[i]=1;

    // Se il numero è ancora marcato come primo (non è multiplo dei suoi precedenti)
    if(array[i]==1) {
        // Marco come "non primi" tutti i suoi multipli
        for (j=2; j<=(100/i); j++){
            array[i*j] = 0;
        }
    }

    printf( "\nNumeri primi < 100: " );

    for (i=1;i<100;i++){
        if(array[i]==1)
            printf("\t%d", i);
    }
    return 0;
}
```

meglio:

j<=(100/i) && j*i<100