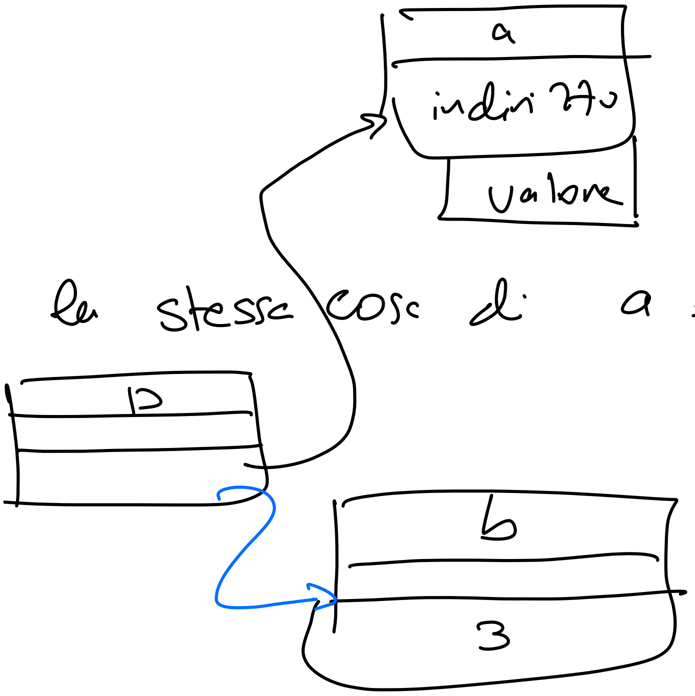
Puntatori e Array 2D

```
int     a;
int*    p;
p = &a;
 *p = 2;
```

la stessa cosa di a = 2

```
int  b;
p = &b;
 *p = 3;
```

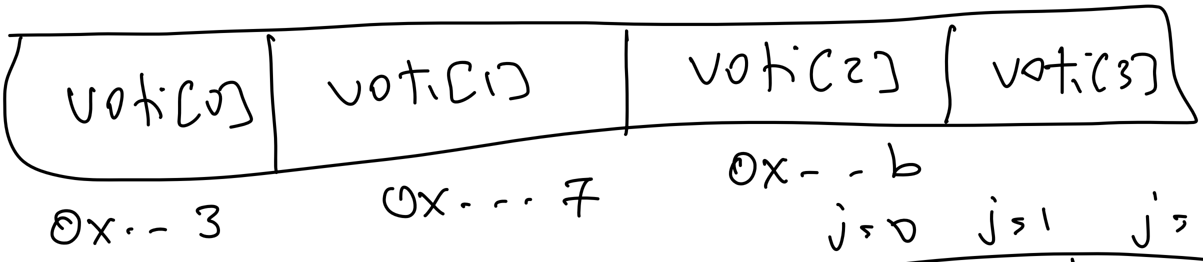


```
int   voti[10];

print("*voti = %d\n", *voti);
```

$*$ voti $\equiv$ voti[0]

(voti+3)   &voti[3].

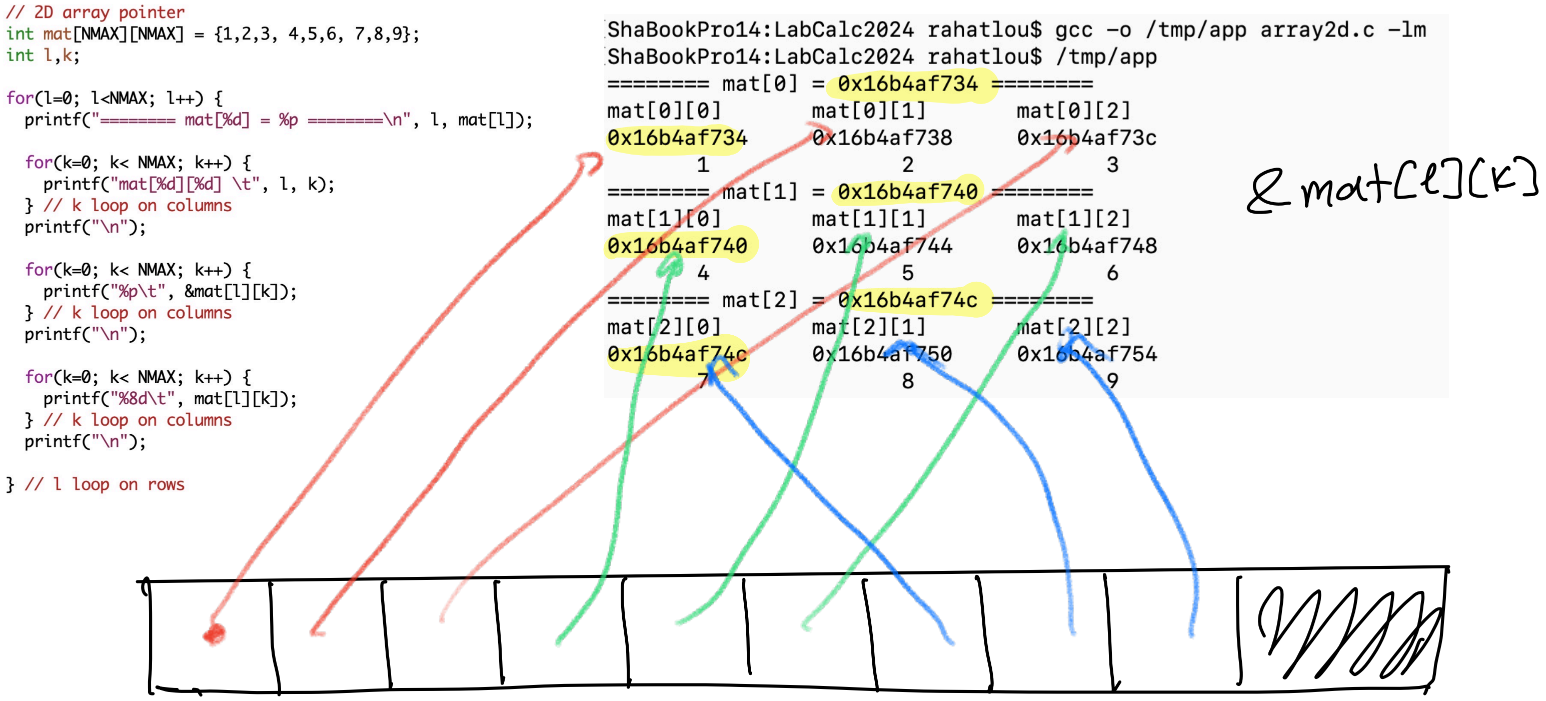
```
*(voti+4) =  29;
  voti[4] =  29
```



```
int   mat[3][3]

mat[1][2]
```

	j=0	j=1	j=2
i=0	1	2	3
i=1	4	5	6
i=2	7	8	9

NMAX = 3;



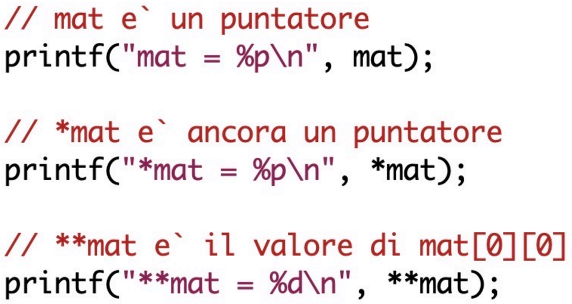
```
// 2D array pointer
int mat[NMAX][NMAX] = {1,2,3, 4,5,6, 7,8,9};
int l,k;

for(l=0; l<NMAX; l++) {
  printf("======== mat[%d] = %p ========\n", l, mat[l]);

  for(k=0; k< NMAX; k++) {
    printf("mat[%d][%d] \t", l, k);
  } // k loop on columns
  printf("\n");

  for(k=0; k< NMAX; k++) {
    printf("%p\t", &mat[l][k]);
  } // k loop on columns
  printf("\n");

  for(k=0; k< NMAX; k++) {
    printf("%8d\t", mat[l][k]);
  } // k loop on columns
  printf("\n");

} // l loop on rows
```

```
ShaBookPro14:LabCalc2024 rahatlou$ gcc -o /tmp/app array2d.c -lm
ShaBookPro14:LabCalc2024 rahatlou$ /tmp/app
======== mat[0] = 0x16b4af734 ========
mat[0][0]       mat[0][1]       mat[0][2]
0x16b4af734     0x16b4af738     0x16b4af73c
       1               2               3
======== mat[1] = 0x16b4af740 ========
mat[1][0]       mat[1][1]       mat[1][2]
0x16b4af740     0x16b4af744     0x16b4af748
       4               5               6
======== mat[2] = 0x16b4af74c ========
mat[2][0]       mat[2][1]       mat[2][2]
0x16b4af74c     0x16b4af750     0x16b4af754
       7               8               9
```

mat[0] → & mat[0][0]

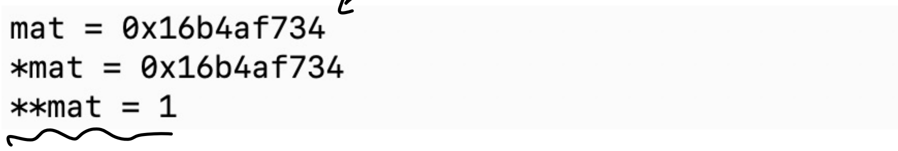
mat[1] → & mat[1][0]

mat[2] → & mat[2][0]



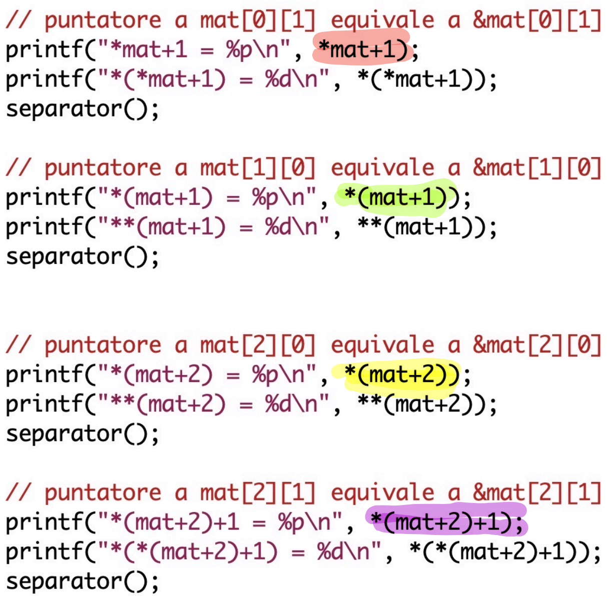
```
// mat e` un puntatore
printf("mat = %p\n", mat);

// *mat e` ancora un puntatore
printf("*mat = %p\n", *mat);

// **mat e` il valore di mat[0][0]
printf("**mat = %d\n", **mat);
```

& mat[0][0]

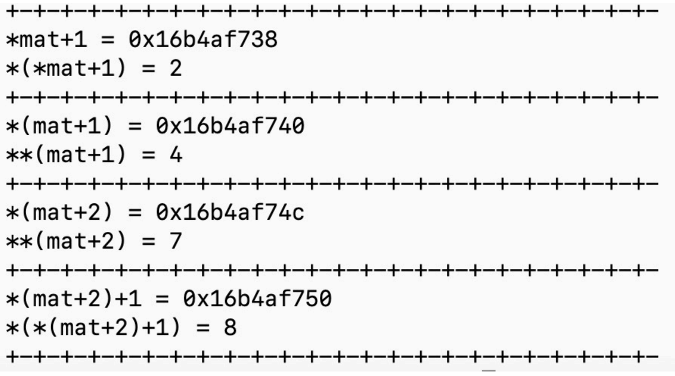


```
mat = 0x16b4af734
*mat = 0x16b4af734
**mat = 1
```

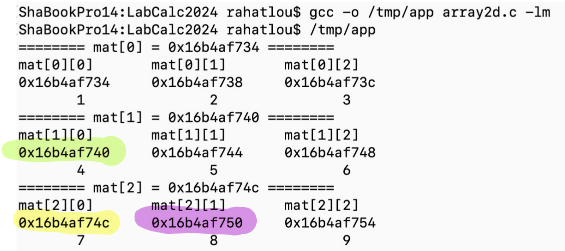
*mat punta a riga Ø
+1 : colonna 1



```
// puntatore a mat[0][1] equivale a &mat[0][1]
printf("*mat+1 = %p\n", *mat+1);
printf("*(*mat+1) = %d\n", *(*mat+1));
separator();

// puntatore a mat[1][0] equivale a &mat[1][0]
printf("*(mat+1) = %p\n", *(mat+1));
printf("**(mat+1) = %d\n", **(mat+1));
separator();


// puntatore a mat[2][0] equivale a &mat[2][0]
printf("*(mat+2) = %p\n", *(mat+2));
printf("**(mat+2) = %d\n", **(mat+2));
separator();

// puntatore a mat[2][1] equivale a &mat[2][1]
printf("*(mat+2)+1 = %p\n", *(mat+2)+1);
printf("*(*(mat+2)+1) = %d\n", *(*(mat+2)+1));
separator();
```



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
*mat+1 = 0x16b4af738
*(*mat+1) = 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
*(mat+1) = 0x16b4af740
**(mat+1) = 4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
*(mat+2) = 0x16b4af74c
**(mat+2) = 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
*(mat+2)+1 = 0x16b4af750
*(*(mat+2)+1) = 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```



```
ShaBookPro14:LabCalc2024 rahatlou$ gcc -o /tmp/app array2d.c -lm
ShaBookPro14:LabCalc2024 rahatlou$ /tmp/app
======== mat[0] = 0x16b4af734 ========
mat[0][0]       mat[0][1]       mat[0][2]
0x16b4af734     0x16b4af738     0x16b4af73c
       1               2               3
======== mat[1] = 0x16b4af740 ========
mat[1][0]       mat[1][1]       mat[1][2]
0x16b4af740     0x16b4af744     0x16b4af748
       4               5               6
======== mat[2] = 0x16b4af74c ========
mat[2][0]       mat[2][1]       mat[2][2]
0x16b4af74c     0x16b4af750     0x16b4af754
       7               8               9
```

int voti[1000][10000]

cella 223, 3431

$$*(*(voti + \underbrace{223}_{righe}) + \underbrace{3431}_{colonne}) = 18$$

voti[223][3431]

int rubik[3][3][3];

((*(rubik + 2) + 1) + 2)

cella piano 2: riga 1, colonne 2

Funzioni

x = sqrt(y);

stampaMatrice(mat);

$x \in [a,b]$ x = a + (b-a) * lrand48()/RAND_MAX;

x = genera(a,b)

gioco = dado(6) $i \in [1,6]$

dd = dado(32) $\in [1,32]$

moneta = generaMoneta()

Funzione: insieme di istruzioni (algoritmo) input: dati di partenze

output: dati o azione da compiere

Funzioni già usate:

```
printf( -- )
sqrt( )
sin (x)
lrand48()
```

```
tipo      nome(argomenti)
          fopen(•, •)
          time( )
          srand48( time(0) );

sqrt( x*x )

sin( sqrt(M_PI) );

printf("%d \n", x*2);

printf("x = %lf \n", sqrt(y) );

y = rad Bibilon (x);

theta = mypi();
```