int, float, double, char

(x, y, z) $(t, \vec{x})$ (t, x, y, z)

(v_1, v_2, v_3)

```
float x, vx, y, vy;

float x1, x2, x3, - - , x99;
x1 = -1;
x2 = -1;
x3 = 0.;
:
!
```

Array in C

```
double posizione[3];
double g = 9.8;
```

Array 1-dimensionale di lunghezza 3

tipo nome_var [lunghezza]

pos: Zione

pos:Zione[0]	pos:Zione[1]	pos:Zione[2]

```
float x[4],
```

x[0]	x[1]	x[2]	x[3]

```
x[0] = 0;
x[2] = -2.2;
x[1] = 1;
x[3] = - M_PI;
```

x[i] i = 0, 1, 2, 3

```
int dati[100];   i = 0, - - -, 99
int d1, d2, d3, - — , d100;
```

```
for(i=0; i<100; i++) {
        dati[i] = 2;
}
```

```
d1 = 2;
d2 = 2;
 :
 :
d100 = 2;
```

$\vec{v}$ v_x, v_y

```
double v[3];
v[i] = 0;
```

```
int pippo = 23;
dati[pippo] = 0;
```

```
int j = 100;
dati[j] = 0;
```

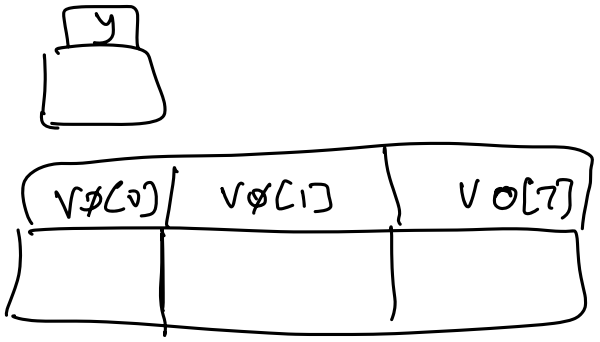
dati[0]	dati[1]	--	dati[99]

```
#define LEN 3
int main() {
    double pos[LEN], vel[LEN], v0[LEN];
    v0[0] = --
    v0[1] = --
    v0[2] = --
  for(int i=0; i<LEN; i++) {
        v0[i] = --i;
  }
```

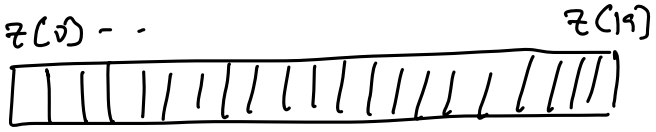
input dell'array

```
    printf("inserisci v0(x): ");
    scanf("%lf", &v0[0]);
    scanf("%lf", &y);
```

double y;

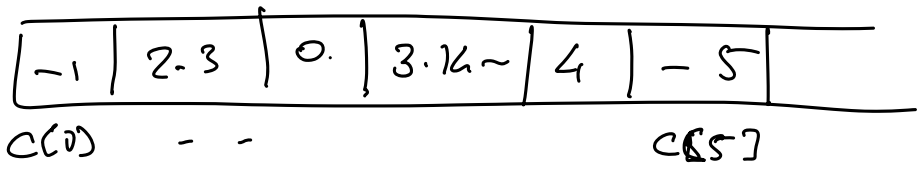


double z[20] = {0};



double w[4] = 0; ERRORE

double c[] = {-1, 2.3, 0., M-PI, 4, -5};



c[1] = - -

c[2] = - -

```
for(int i=0; i<6; i++){
    printf("c[%d] = %lf\n", i, c[i]);
}
```

c[0] = - -

c[1] = - -

printf("c[%d] = %lf\n", i+1, c[i]);

c[1] = - -

c[2] = - -

:

c[6] = - -

```
for(int i=1; i<=6; i++){
    C[i] = --
```

C[1] ✓
C[2] ✓
C[3] ✓
C[4] ✓
C[5] ✓
C[6] NON ESISTE
C[0] mai usato.

```
for (int i = 0; i < 3; i++) {
    printf("inserisci: V(id) = ", i);
    scanf("%lf", &V[i]);
}
```

```
for (int i = 0; i < 3; i++) {
    do {
        printf("inserisci: V(id) = ", i);
        scanf("%lf", &V[i]);
        if(V[i] < 0) printf("error: val. negativa\n");
    } while (V[i] < 0);   // acquisizione V[i]
} // ciclo su elementi V[i]
```

Sintassi per array statico.
lunghezza fissata a compilazione:

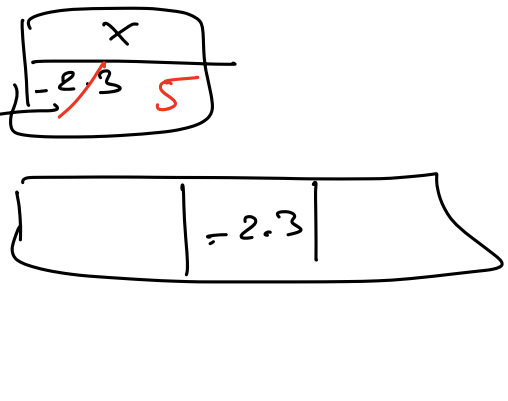
```
int V[20];
```

```
int n;
printf("inserisci lunghezza array: ");
scanf("%d", &n);
double dati[n];
```

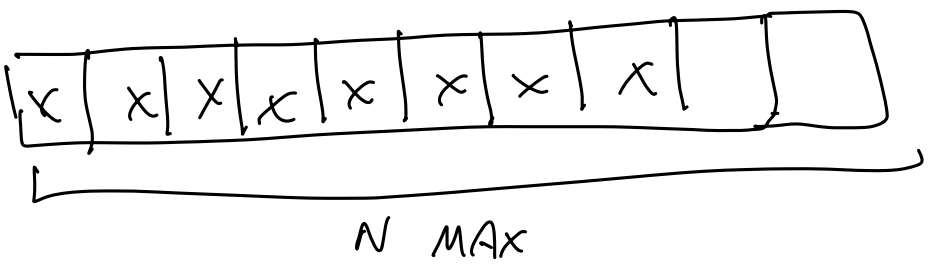
ERRORE

lunghezze array fissate a compil. prima di runtime

```
float x = -2.3;
float V[3];
V[i] = x;
x = 5;
```



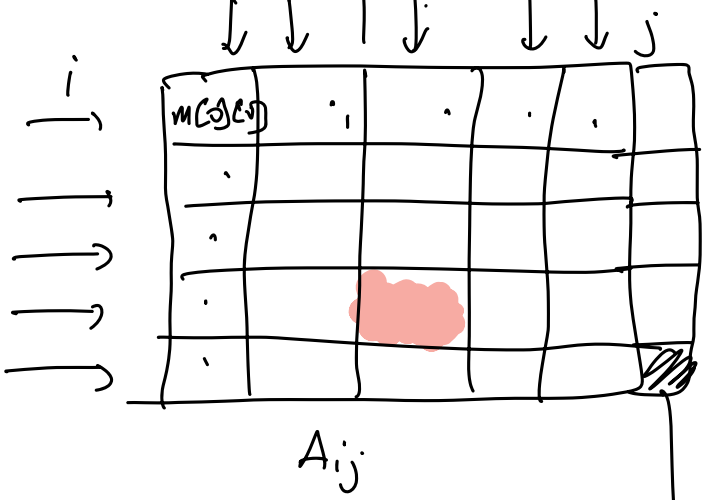

```
#define NMAX 10
int main() {
    double d[NMAX] = {0};
    int n;
    do {
        printf("inserire lunghezza n<%d: ", NMAX);
        scanf("%d", &n);
    } while (n<=0 || n>NMAX);

    for(i=0; i<n; i++) {
        printf("inserisci elemento i: ");
        scanf("%lf", &d[i]);
    }
```



n = 3
n = 8

fecicle per uso di array apparentemente di lungh. vcn.

array / vettore



double mat[5][6];

array bi-dimensionale

m[3][2] → colonna 2

↓ riga 3

partendo da zero.

m[0][0] — - —

m[4][5]
i j

#define NR 7.

#define NC 8

int main() {

float mat[NR][NC];

for(int i = 0; i < NR; i++) {

for(int j = 0; j < NC; j++) {

mat[i][j] = 3.9;

printf("i=%d, j=%d\n", i, j);

} // ciclo colonne, j

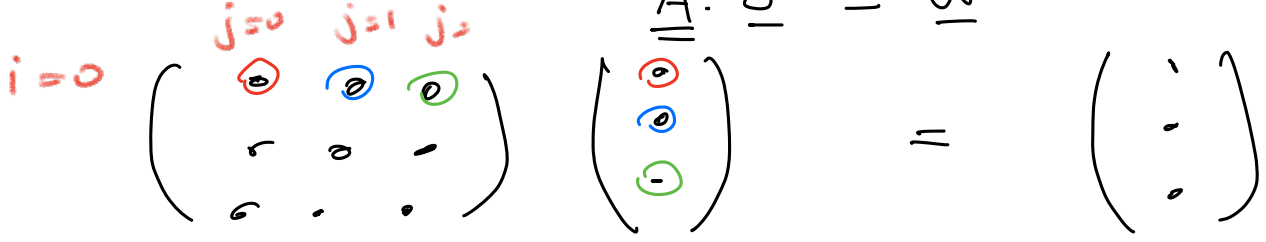
} // ciclo righe, i

mat array bidimensionale

i	j
0	0
0	1
0	2
0	3
0	4
0	5
0	6
0	7
1	0
1	1
1	2

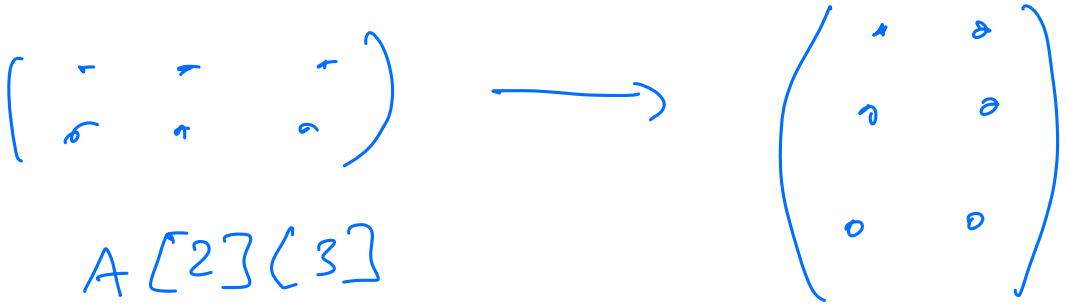
printf ("%lf \n", mat[3][2]);

$A \cdot \vec{v} = \vec{w}$




```
double A[3][3],
   double w[3];
int i,j;
for(i=0; i<3; i++){
      w[i] = 0.;
      for(j=0; j<3; j++){
        w[i] += A[i][j] * V[ ] ;
        printf("A[%d][%d] = %lf\t", i,j, A[i][j]);
      }
      printf("\n");
      pintf("w(%d) = %lf \n", i, w[i]);
}
```

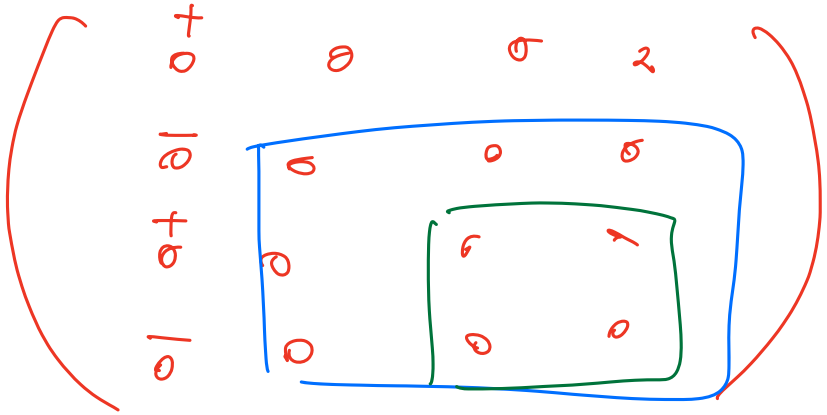
i	j
0	0
0	1
0	2
1	0
1	1
1	2



A[2][3]

B[3][2]

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



if(i%2){

resto divisione per 2

}