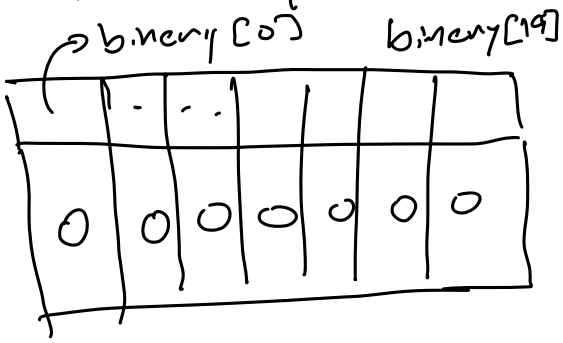
```c
#include <stdio.h>
#include <math.h>

#define NBIT 20

int main() {

   int in, cifra=0, resto;

  do{
    printf("Inserisci numero positivo < %d in base 10: ", (int)pow(2,NBIT));
    scanf("%d", &in);
  } while( in <= 0 || in >= pow(2, NBIT) );

  int in10 = in;

  printf("base 10: %d \t base 2: ", in10);
  do {

   resto = in % 2;
   // stampa del resto come la cifra in binario
   printf("%1d", resto);

   in /= 2;
   cifra++;

  } while (in != 0);

  printf("\n");

  printf("questa conversione errata scrive le cifre al contrario!\n");
}
```

```
shamacmini:material rahatlou$ gcc -o /tmp/app binaryErr.c
shamacmini:material rahatlou$ /tmp/app
Inserisci numero positivo < 1048576 in base 10: 8
base 10: 8       base 2: 0001
questa conversione errata scrive le cifre al contrario!
```

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

#define NBIT 20

int main() {

   int in, cifra=0, resto;
   int binary[NBIT] = {0};

  do{
    printf("Inserisci numero positivo < %d in base 10: ", (int)pow(2,NBIT));
    scanf("%d", &in);
  } while( in <= 0 || in >= pow(2, NBIT) );

  int in10 = in;

  do {

   resto = in % 2;
   printf("%3d-esima cifra: %d \t moltiplica 2 ^%3d (%8.0f)\n",cifra+1, resto, cifra,pow(2,cifra));

   binary[cifra] = resto;
   in /= 2;
   cifra++;

  } while (in != 0);

  printf("base 10: %d \t base 2: ", in10);
  for(int i = cifra-1; i>= 0; i--) {
    printf("%1d", binary[i]);

  }
  printf("\n");

}
```

array statici C
lunghezza fissata alla compilazione



input = 8

	div	r
8/2	4	0
4/2	2	0
2/2	1	0
1/2	0	1

cifra	binary[cifra]	resto
0	0	0
1	0	0
2	0	0
3	1	1

Fine ciclo cifre = 4

ciclo i = 3 , i >= 0

cifra-1

printf("%d", binary[cifre]);

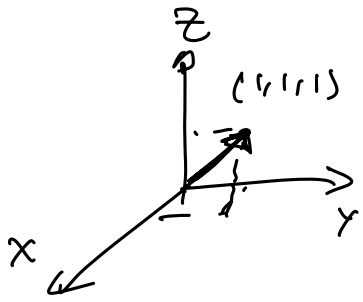
i = 3	⟶	1
i = 2		10
i = 1		100
i = 3		1000

```
shamacmini:material rahatlou$ /tmp/app
Inserisci numero positivo < 1048576 in base 10: 8
  1-esima cifra: 0       moltiplica 2 ^  0 (       1)
  2-esima cifra: 0       moltiplica 2 ^  1 (       2)
  3-esima cifra: 0       moltiplica 2 ^  2 (       4)
  4-esima cifra: 1       moltiplica 2 ^  3 (       8)
base 10: 8       base 2: 1000
```

```
shamacmini:material rahatlou$ /tmp/app
Inserisci numero positivo < 1048576 in base 10: 12233
  1-esima cifra: 1       moltiplica 2 ^  0 (       1)
  2-esima cifra: 0       moltiplica 2 ^  1 (       2)
  3-esima cifra: 0       moltiplica 2 ^  2 (       4)
  4-esima cifra: 1       moltiplica 2 ^  3 (       8)
  5-esima cifra: 0       moltiplica 2 ^  4 (      16)
  6-esima cifra: 0       moltiplica 2 ^  5 (      32)
  7-esima cifra: 1       moltiplica 2 ^  6 (      64)
  8-esima cifra: 1       moltiplica 2 ^  7 (     128)
  9-esima cifra: 1       moltiplica 2 ^  8 (     256)
 10-esima cifra: 1       moltiplica 2 ^  9 (     512)
 11-esima cifra: 1       moltiplica 2 ^ 10 (    1024)
 12-esima cifra: 1       moltiplica 2 ^ 11 (    2048)
 13-esima cifra: 0       moltiplica 2 ^ 12 (    4096)
 14-esima cifra: 1       moltiplica 2 ^ 13 (    8192)
base 10: 12233   base 2: 10111111001001
```

int binary[20]; array unidimensionale con 20 elementi; lunghezza = 20

double pos[3];



double origine[3] = {0, 0, 0};

double versore[3] = {1, 1, 1};

double versorex[3] = {1, 0, 0};

double versorey[3];

~~versorey = {0, 1, 0};~~

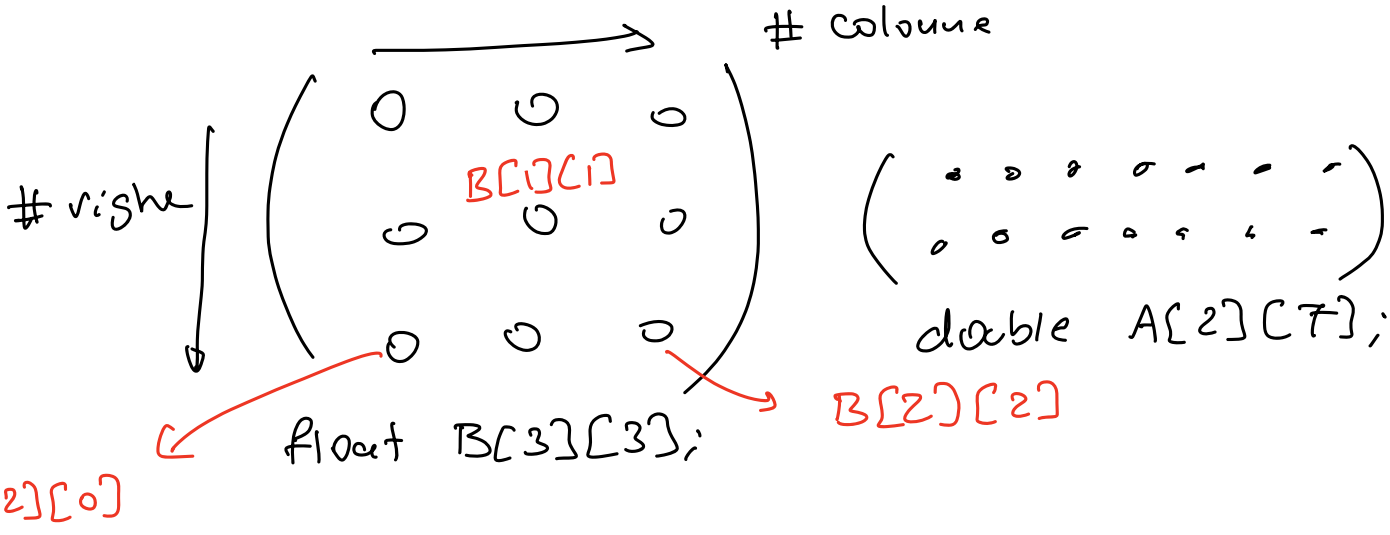
versory[0] = 0;

versorey[1] = 1;

versorey[2] = 0;

nome array sempre con operatore [.]

int matricola[170];



int scacchiera[8][8]

righe

colonne.

```
double m[2][2] = {1, 0, 0, 1};
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```
double v1[3]
double v2[3];
```

$|\vec{v}_i| = \sqrt{\sum_i v_i^2}$

$\vec{v}_1$, $\vec{v}_2$, m

prod. scalare $\vec{v}_1 \cdot \vec{v}_2$

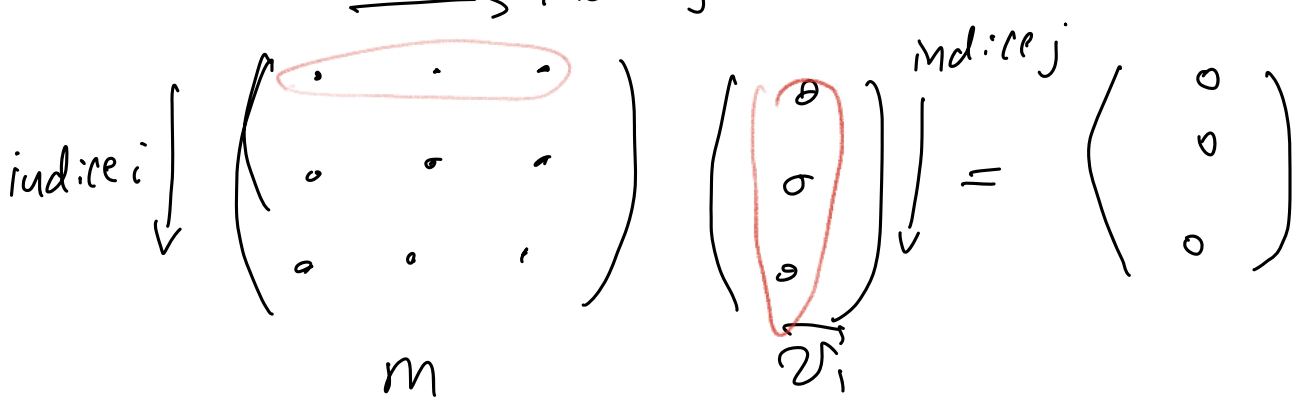
```
double modV1 = 0;
for(int i = 0; i < 3; i++) {
        modV1 += V1[0] * V1[0];
}
modV1 = sqrt(modV1);

double prod = 0;
for(int i = 0; i < 3; i++) {
        prod += V1[i] * V2[i];
}
```

$V1[0] \cdot V2[0] + V1[1] \cdot V2[1] + V1[2] \cdot V2[2]$

$\vec{v}_3 = \vec{v}_1 \times \vec{v}_2 =$

$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ V_1[0] & V1[1] & V1[2] \\ V2[0] & V2[1] & V2[2] \end{vmatrix}$$

```
v3[0] =
```



```
int i,j;
double v3[3] = {0};
for(i=0; i<3; i++) {
    double p=0;
    for(j=0; j<3; j++) {
        p += m[i][j] * v1[j];
    } // ciclo j
    printf("v3[%d] = %f\n", i);
    v3[i] = p;
} // ciclo i
```

```
// array statico di lunghezza NMAX
double v1[NMAX] = {1 , 2.5, -1.5};
double v2[NMAX] = { -1, 2, 3.5};

// matrice NMAX x NMAX
double mat[NMAX][NMAX] = { 1, 0, -1, 0, 1, 0, 2, 0, 1 };

int i,j;

// ora stampa i valori nel vettore
printf("=== ecco i valori salvati:\n");
for(i = 0; i < NMAX; i++) {
  printf("v1[%d] = %.3f \t v2[%d] = %.3f\n", i, v1[i], i, v2[i]);
} // ciclo output
printf("\n\n");

for(i = 0; i < NMAX; i++) {
  for(j = 0; j < NMAX; j++) {
    // \t equivale al tasto TAB
    printf("mat[%d][%d] = %.3f \t", i, j, mat[i][j]);
  } // ciclo j
  // finita la stampa della riga si va a capo
  printf("\n");
} // ciclo i

//prodotto scalare
double scalProd = 0.;
for(i = 0; i < NMAX; i++) {
    scalProd += v1[i]*v2[i];
} // ciclo output
printf("\nv1 * v2 = %.3f\n", scalProd);
```

```
#define NMAX 3
```

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{pmatrix}$$

```
    // v3 = mat x v1
    double v3[NMAX]={0};

    //prodotto  mat x v1
    printf("\nv3[] = mat[][] x v1[] \n");
    for(i = 0; i < NMAX; i++) {
      double p = 0.;
      for(j = 0; j < NMAX; j++) {
        p += mat[i][j] * v1[j];
      } // ciclo j colonne mat

      v3[i] = p;
      printf("v3[%d] = %.3f\n", i, v3[i]);

    } // ciclo i righe mat
```

```
shamacmini:material rahatlou$ gcc -o /tmp/app array2.c
shamacmini:material rahatlou$ /tmp/app
=== ecco i valori salvati:
v1[0] = 1.000    v2[0] = -1.000
v1[1] = 2.500    v2[1] = 2.000
v1[2] = -1.500   v2[2] = 3.500


mat[0][0] = 1.000       mat[0][1] = 0.000       mat[0][2] = -1.000
mat[1][0] = 0.000       mat[1][1] = 1.000       mat[1][2] = 0.000
mat[2][0] = 2.000       mat[2][1] = 0.000       mat[2][2] = 1.000

v1 * v2 = -1.250

v3[] = mat[][] x v1[]
v3[0] = 2.500
v3[1] = 2.500
v3[2] = 0.500
```