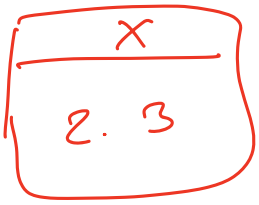
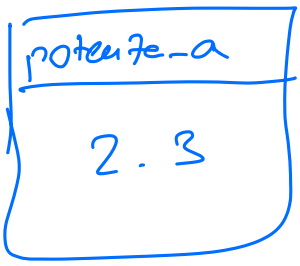
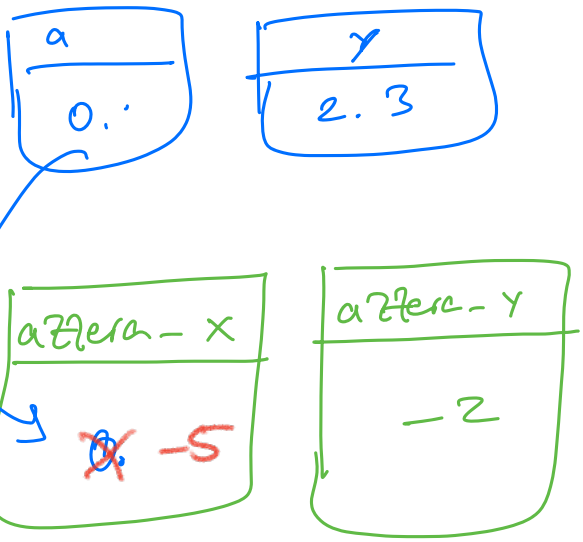
Funzioni: passaggio per valore e per puntatore

```
double potenza(double, int);

int main() {
    double x = 2.3;
    y = potenza(x, 3);
}

double potenza(double a, int n) {
    double r = 1;
    for(int i=0; i<n; i++) {
        r *= a;
    }
    return r;
}
```





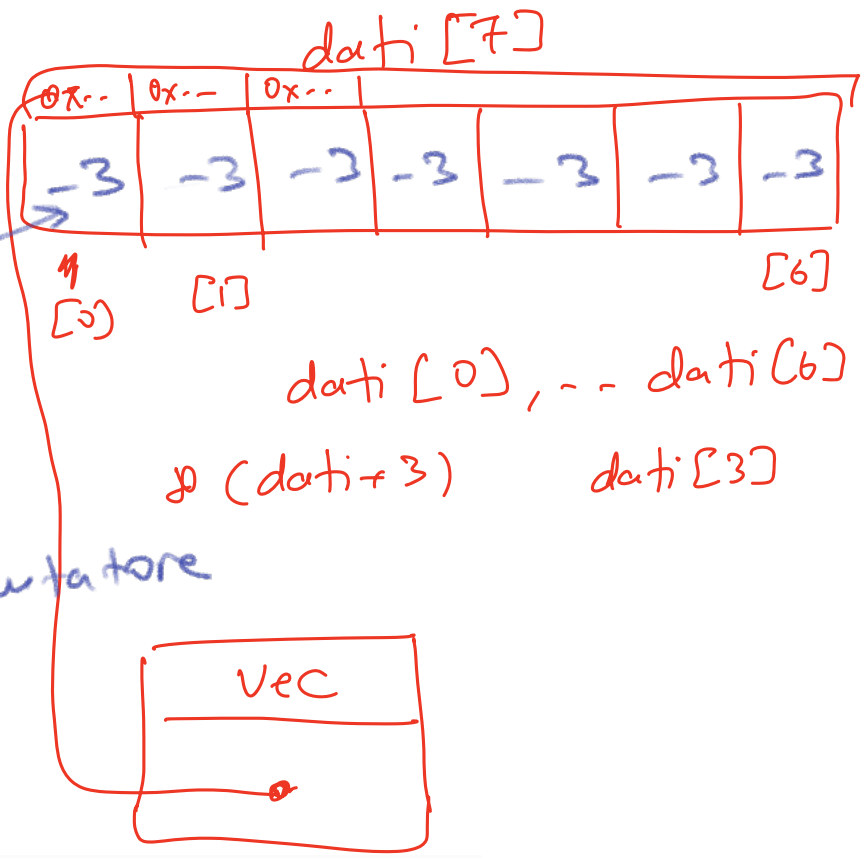

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  void azzera(double);
5
6  int main() {
7
8      double a = 0., y = 2.3;
9      printf("inizio main: a = %f , y = %f\n", a, y);
10     azzera(a);
11     printf("fine main: a = %f , y = %f\n", a, y);
12 }
13
14 void azzera(double x) {
15     double y = -2.;
16     printf("inizio azzera: x = %f, y = %f\n", x , y);
17     x = -5.;
18     printf("fine azzera: x = %f, y = %f\n", x , y);
19 }
```




```
inizio main: a = 0.000000 , y = 2.300000
inizio azzera: x = 0.000000, y = -2.000000
fine azzera: x = -5.000000, y = -2.000000
fine main: a = 0.000000 , y = 2.300000
```

passaggio per valore

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

// stampa elementi array
void printVec(double*, int);

// assegna un valore a tutti gli elementi di un array
//void modVec(double, double*, int);
void modVec(double, double [], int);

int main() {

    double dati[7] = {0};
    printVec(dati, 7);
    modVec( -3., dati, 7);
    printVec(dati, 7);

}

void printVec(double* vec, int lun){
    printf("printVec:\n");
    for(int i=0; i<lun; i++){
        printf("%.4f\t", *(vec+i) );
    }
    printf("\n");
}

//void modVec(double val, double* vec, int lun) {
void modVec(double val, double vec[], int lun) {
    printf("modificando il vettore in modVec()\n");
    for(int i=0; i< lun; i++){
        *(vec+i) = val;
    }
}
```




```
printVec:
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
modificando il vettore in modVec()
printVec:
-3.0000 -3.0000 -3.0000 -3.0000 -3.0000 -3.0000 -3.0000
```

```c
double media(double* V, int n);

double voti = {29., 30., 27., 29., 21};
double m = media(voti, 5);

double media(double* V, int n) {
    double s = 0;
    for(int i=0; i<n; i++) {
        s += V[i];
        V[i] = -1; // errore malizioso
    }
    s /= n;
    return s;
}
```

```
double modulo(double v[], int n) {
    double s=0;
    for(int i=0; i<n; i++) {
        s += v[i] * v[i];
    }
    return sqrt(s);
}
```

file.c

dichiarazione funzioni:

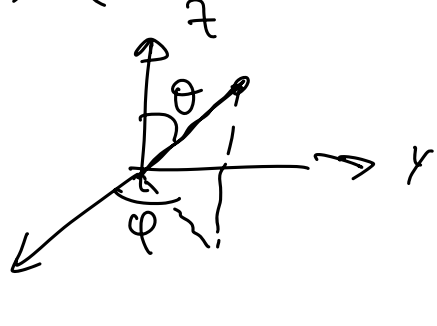
```
int main(){
}
```

implementazione funzioni:

ambito visibilità

```
int main() {
    for(int i=0; i<3; i++){
    }
    printf("i = %d", i);
}
```

```
double theta(double v[], int n) {
double phi(double* v, int n) {
}
double prodscalare(double* v1, double* v2, int n)
```



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

// stampa elementi array 2D
// bisogna specificare dimensione e lunghezza max di array
void printMat(double [7][4], int, int);

// assegna un valore a tutti gli elementi di un array
void assegna(double, double [7][4], int, int);


int main() {

    double mat[7][4] = {0};
    printMat(mat, 7, 4);
    assegna( 3.14, mat, 7, 4);
    printMat(mat, 7, 4);

}

void printMat(double m[7][4], int righe, int col){
    printf("printMat:\n");
    for(int i=0; i<righe; i++){
        for(int j=0; j<col; j++) {
            printf("%.4f\t", m[i][j] );
        }
    printf("\n");
    }

}

void assegna(double val, double m[7][4], int nr, int nc) {
    printf("modificando valori matrice in assegna()\n");
    for(int i=0; i< nr; i++){
        for(int j=0; j<nc; j++){
            *(*(m+i)+j) = val;
        }
    }
}
```

```
printMat:
0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000
modificando valori matrice in assegna()
printMat:
3.1400  3.1400  3.1400  3.1400
3.1400  3.1400  3.1400  3.1400
3.1400  3.1400  3.1400  3.1400
3.1400  3.1400  3.1400  3.1400
3.1400  3.1400  3.1400  3.1400
3.1400  3.1400  3.1400  3.1400
3.1400  3.1400  3.1400  3.1400
```

```c
void generaMat(a, b, m, nr, nc);
```

```c
void generaMat(double a, double b, double m[3][3], int nr, int nc){
    for(int i=0; i<nr; i++){
        for(int j=0; j<nc; j++){
            m[i][j] = a + (b-a)*(rand()/RAND_MAX);
        }
    }
}
```

per generare una matrice 3x3 con valori in [-2,1]

```c
double mat[3][3];
generaMat(-2, 1., mat, 3, 3);
```