

## Funzioni:

```
tipo nome(arg1, arg2, ..., argN) {  
    tipo qualcosa  
    // corpo funzione  
    return qualcosa;  
}
```

```
double mypi() {  
    double res;  
    ===  
    return res;  
}
```

valore di output  
della funzione mypi()

```
int main() {
```

```
    double x;  
    x = mypi();  
    double y = sqrt( mypi() );  
    return 0;  
}
```

./eseguibile

script shell:

```
x = `./eseguibile`
```

Funzione dado()

```
int dado(int facce) {  
    int res = rand48() % facce;  
    res += 1;      res = res + 1;     $res \in [1, facce]$   
    return res;  
}
```

$\in [0, facce-1]$

la stessa cosa

```
int dado(int facce) {  
    return (rand48() % facce) + 1;  
}
```

prog. C

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

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

```
#include <time.h>
```

```
int dado(int); // dichiarazione di dado()
```

```
int main() {  
    int facce;  
    srand48(time(0));  
    printf("n facce dado: ");  
    scanf("%d", &facce);  
    res = dado(facce);
```

main - res
0x--1b
<del>3</del> 3

Chiamare funzione dado()

```
    printf("dado: %d\n", dado(6));  
    printf("res: %d\n", res);  
    printf("dado: %d\n", dado(6));
```

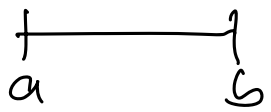
main - facce
0x--13



```

double genera (double x, double y) {
    return x + ((double) rand48() / RAND_MAX) * (y - x);
}

```



```

int main() {
    double x = -1.1, y = 1.2;
    double z = genera(x, y);
}

```

unico modo per passare valore di una variabile  
 è tramite argomenti della funzione;

Esempio acquisizione.  $0 < L < d$

```

double prendiDouble (double, double);
int prendiInt (int, int);
int main() {
    double L = prendiDouble(0, d);
    int nexp = prendiInt(0, 100);
}

```

```

double prendiDouble (double xinf, double xsup) {
    double x;
    do {
        printf("inserisci x tra %lf e %lf", xinf, xsup);
        printf("x in [%lf, %lf]:", xinf, xsup);
        scanf("%lf", &x);
    } while (x < xinf || x > xsup);
}

```

return x;

}

int prendiInt ( int xinf, int xSup ) {

int x;

do  
while

}

int main () {

double d = prendiDouble ( 0, 5 ); // in con.

double L = prendiDouble ( 0, d ); // in con

int dado ( int facce ) {

probabilità =  $1/\text{facce}$

double x = (double) rand48() / RAND\_MAX;

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

if ( x > (double) i / facce && x < (double) (i+1) / facce ) {

return i+1;

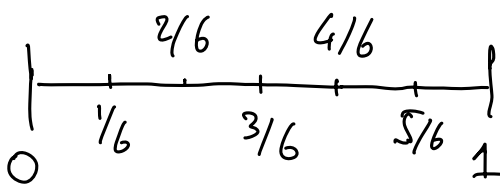
}

}

}

$i \in [0, 5]$ .

$x \in [i/\text{facce}, (i+1)/\text{facce}]$



creare func.c : con tutte funzioni.  
no main. } 900  
funzioni:

func.h : solo dichiarazioni.

gcc -c func.c : crea func.o

prog.c

```
#include <func.h>
```

```
int main()
```

```
{  
  //  
  //  
  //  
}
```

gcc -o app.exe prog.c func.o -lm

gcc -o app.exe prog.c func.c -lm.

```
#include < >
```

```
// dichiarazioni funzione
```

```
int dado(int);
```

```
int main() {
```

```
  //  
  //  
  //  
}
```

```
int dado(int n) {
```

```
  //  
  //  
  //  
}
```

prog.c

gcc -o app.exe prog.c -lm

L = prendiDouble("lunghezza", 0, d)

d = prendiDouble("distanza", 0, 5)

nexp = prendiInt("n Simulazioni", 1, 100)

double prendiDouble(char\* stringa, double x1, double x2) {

==

do {

printf("%s [%i | %i] :", stringa, x1, x2);

==

}

}

Funzioni senza valore di ritorno: void

void benvenuto(char\* mess) {

printf(" programma per calcolo pisreconi");

printf("%s\n", mess);

}

int main() {

benvenuto(" 27 novembre 2024");

}

```
int voti [100];
```

```
int somma = 0;
```

```
for (int i = 0; i < 100; i++) {
```

```
    somma += *(voti+i);
```

```
}
```

```
double media = (double) somma / 100;
```

Vourei:

```
double media = calcolaMedia( voti );
```

voti non conosce la sua lunghezza

Soluzione:

```
double avg = calcolaMedia( voti, 100 );
```

Dichiarazione:

```
double calcolaMedia( int*, int );
```

Implementazione:

```
double calcolaMedia( int* d, int n ) {
```

```
    int somma = 0;
```

```
    for (int i = 0; i < n; i++) {
```

```
        somma += *(d+i); // usando puntatore
```

```
oppure    somma += d[i]; // usando d come array
```

```
}
```

```
    return (double) somma / n;
```



```

oppure double med = (double) somma / n;
return med;
}

```

```
double CalcolaMedia(double* d, int n) {
```

Array come argomento: è richiesto possesso della lunghezza come argomento.

```
#define NEXP 100
```

```
int main() {
```

```
    ==
```

```
    double dati[NEXP] = {0};
```

```
    for (int i = 0; i < NEXP; i++) {
```

```
        dati[i] = genera(0, 1);
```

```
    }
```

```
    double med = CalcolaMedia(dati, NEXP);
```

nessuno impedisce di fare

```
CalcolaMedia(dati, 3) // solo 3 valori
```

```
CalcolaMedia(dati, 1000) // usa valori
                           che non hanno
                           parte dell'array.
```

```
double mediaConfiduce(double* d, int primo, int n) {
```

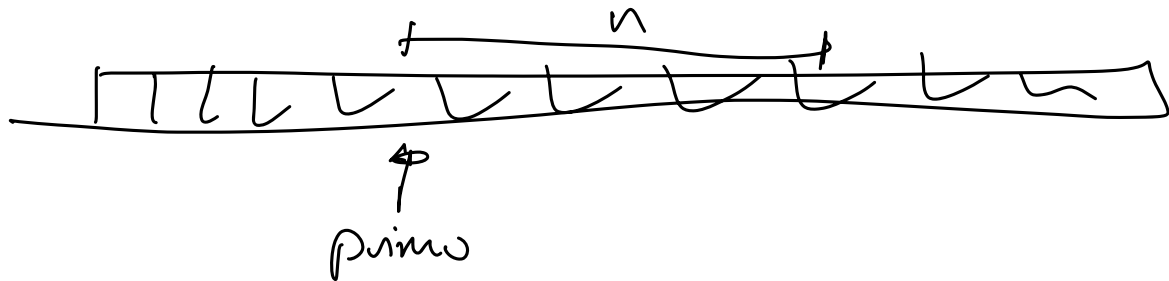
double somme = 0;

for (int i = primo; i < primo + n; i++) {

    somme += x(d + i);

}

return somme / n



void altera(double\* d, int n) {

    for (int i = 0; i < n; i++) {

        d[i] = 0;

    }

}

int main() {

    double d[100] = {1};

    int n = 10;

altera(d, n);