```
double mat[3][3], v[3];

for(int i=0; i<3; i++) {
    v[i] = 0;
    for(int j=0; j<3; j++) {
        mat[i][j] = 0;
    }
}

double w[3] = {-1};
double z[3] = {1, 2, 3};
```

Generazione Numeri Casuali pseudo-casuali

```
#include <stdlib.h>
#include <time.h>
int main() {
    srand48(time(0));
```

```
    int seed = time(0);
    srand48(seed);
```

Inizializza sequenza numeri pseudo-cas. una volta sola per intero programma

```
int n = lrand48();   ∈ [0, RAND_MAX]
```

↳ numero intero

① Generazione moneta / dado a 2 facce

```
int moneta = lrand48()%2;
```

```
int npari, ndispari;
npari = ndispari = 0;
for(int i=0; i< 100000; i++){
    if (rand48()%2){
        ndispari++
    } else {
        npari++;
    }
    if( (rand48()%2) == 1)
}
```

In alternativa

Dado a 6 facce:

```
int nfacce = 6;
int dado = (rand48()% nfacce) + 1
```

~~0, 1, 2, 3, 4, 5~~

1, 2, 3, 4, 5, 6

Errore:

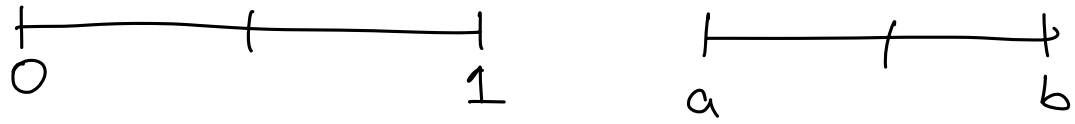
```
rand48() % (nfacce + 1)
```

0, 1, 2, - - , nfacce

```
int Contatore[6];
for(int i=0; i<1000; i++){
    int j = rand48()% nfacce;
    Contatore[j]++;
}
```

```
for(int i=0; i<nfacce; i++){
    printf("frazione %d: %.3lf\n", i, contatore[i]/1000);
}
```

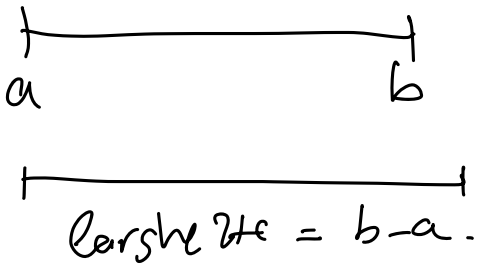
Generare dado a 2 facce



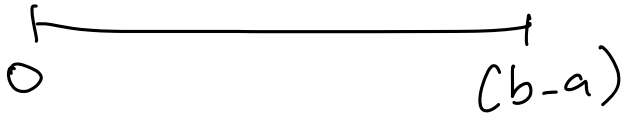

```
double x = (double)rand()/RAND_MAX;
rand()/(double)RAND_MAX;
```

$x \in [0, 1]$

Generare $y \in [a, b]$



$y = x * (b-a)$

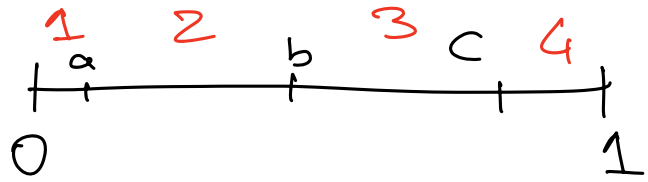


$[0, b-a]$

$y = x * (b-a) + a \quad \in [a, b]$

```
x = rand()/(double)RAND_MAX;
if(x>=0.5){
        npari++;
} else {
        ndispari++;
}
```

H → γ γ̄ 10^3
b b̄ 58%



int contatore [4]

```
. x = (double) rand()/RAND_MAX
  if(x > c) {
      contatore[3]++;
  } else if (x > b) {
      contatore[2]++;
  } else if(x > a) {
      contatore[1]++;
  } else {
      contatore[0]++;
  }
```

```
int v[10];
for(int i=0; i<10; i++) {
    v[10] = rand48() % 100;
}
```

$a = -5 \qquad b = -3$

$b - a = -3 - (-5) = 2.$

$y = x * 2 + (-5) \qquad y \in [-5, -3]$