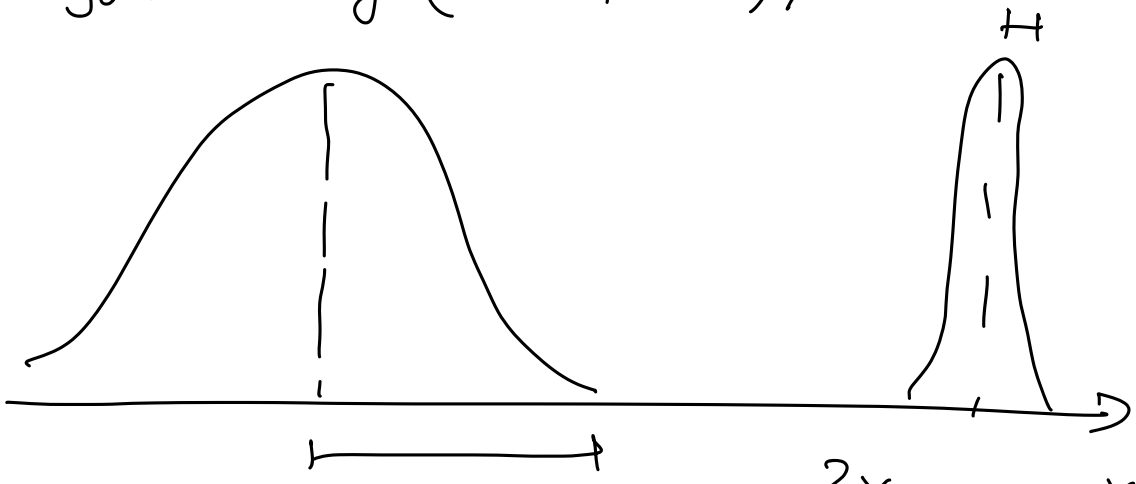
```
int voti[170];
double m = media(voti, 170);
```

Non esiste funzione che ritorni array

voti = media(array, N);

```
void generaArray(double* d, int n) {
    for(int i=0; i<n; i++){
        *(d+i) = (double) rand()%i / RAND_MAX;
        d[i] = --          // la stessa cosa
    }
    // non c'è alcun return qualcosa
}

int main() {
    double dati[100];
    generaArray(dati, 100);
}
```



$\{x_1, \dots, x_N\}$

$\hat{\mu} = \frac{\sum_i^N x_i}{N}$ media aritmetica

$\hat{\sigma}$ $\sqrt{\text{varianza}(x)}$

$$var = \frac{\sum_i (x_i - \bar{x})^2}{N-1}$$

$$= \langle x^2 \rangle - \langle x \rangle^2$$

$$\langle x^2 \rangle = \frac{\sum_i^N x_i^2}{N}$$

30	27	18	28

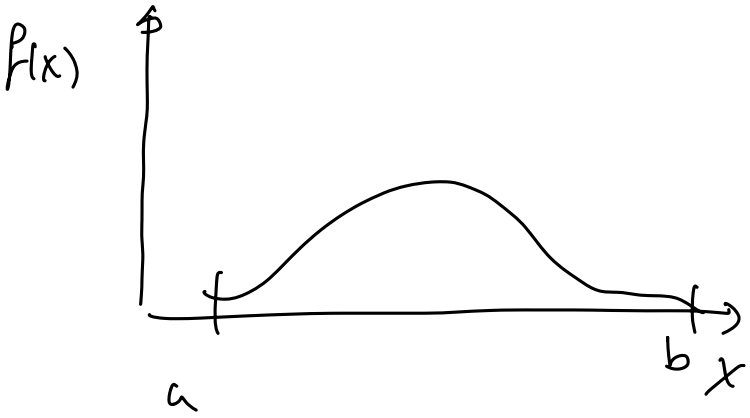
27	28	27	25

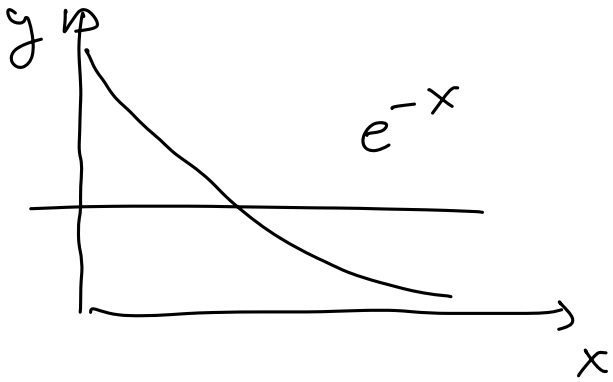
```
int main(){
    double dati[1000] = - ~ ;
    double media, devstd;
    analisi(dati, &media, &devstd, 1000);
    printf("media = %.3f, devstd = %.3f\n", media, devstd);
}
void analisi(double* dati, double* m, double* std, int n){
    double s=0, s2=0;   // *m = 0; *std = 0;
    for(int i=0; i<n; i++){
        s += dati[i];   // Σ_i x_i   // *m += dati[i];
        s2 += (*(dati+i))*(*(dati+i)); // Σ_i x_i^2
    }
    s /= n;   // *m /= n;
    s2 /= n;

    *m = s;
    *std = sqrt(s2);
}
```

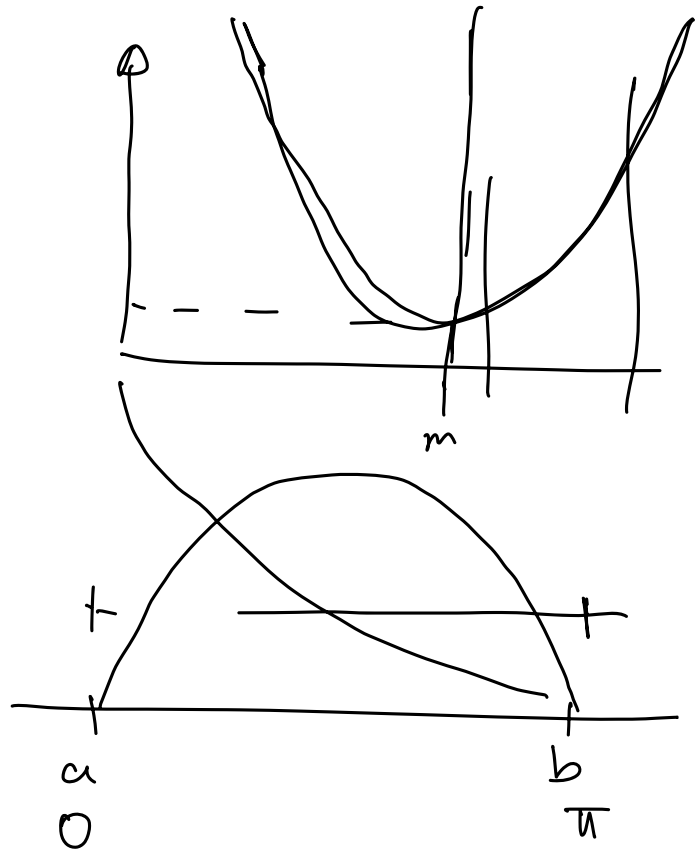
sostituire: s → *m

s2 → *std

Generare num. casuali secondo f(x)







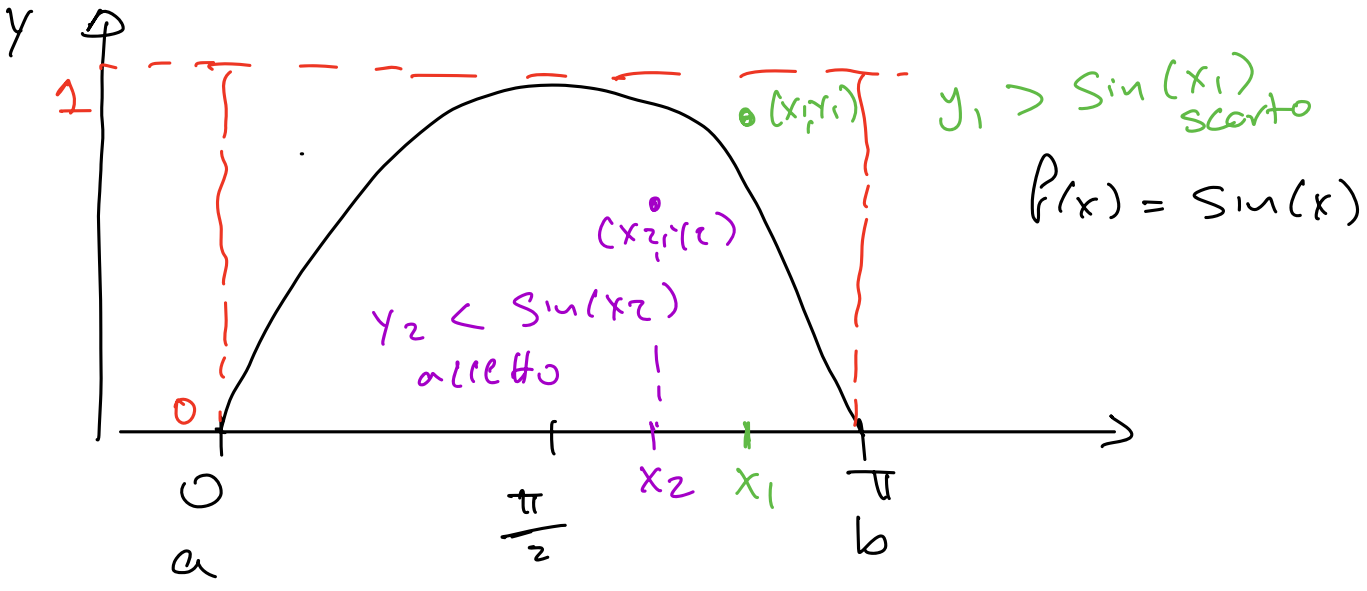
$a + bx + cx^e$

$f(x) = a - bx^3$

$f(x) =$ costante

$f(x) = \sin(x)$

$f(x) = e^{-x}$



$f(x) = \sin(x)$

uniforme: $x = a + (b-a) * (\text{rand}()\ /\ RAND\_MAX)$;

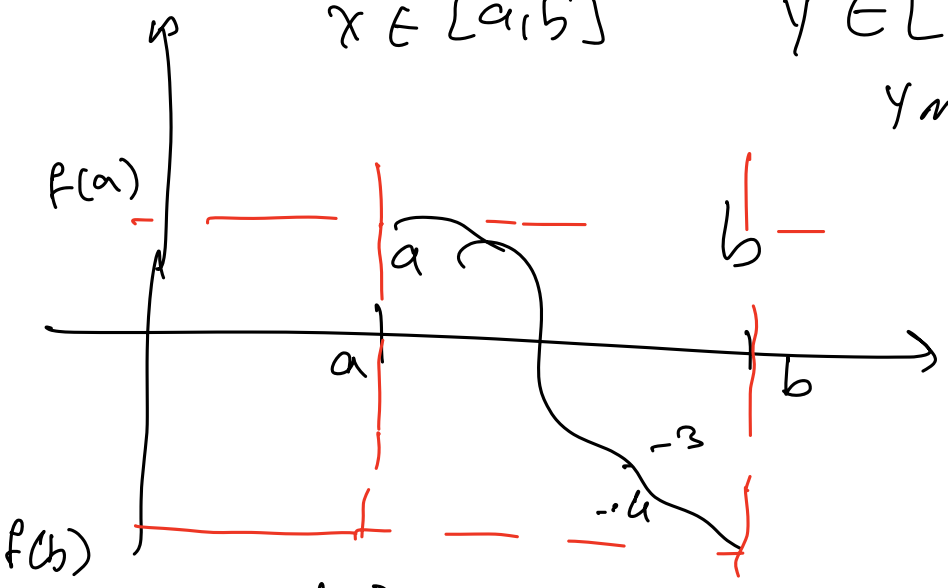
$x \in [a,b]$

max $f(x)$ per $x \in [a,b]$

```
double x,y;
do {
    x = a+(b-c)*(rand()/RAND_MAX;
    y = (double)rand()/RAND_MAX;
} while ( y > sin(x) );
```

$x \in [a,b]$ $\quad y \in [0, y_{max} = 1]$

$y_{max} = \max f(x)$ in $[a,b]$



$x \in [a,b]$.

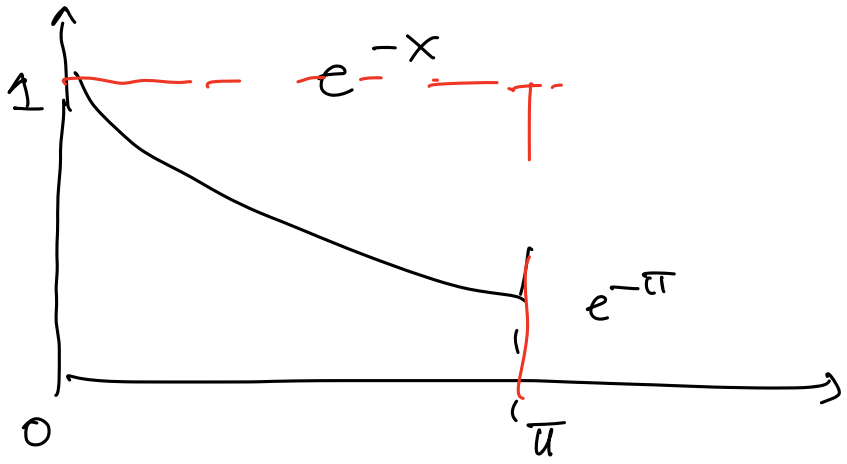
$y \in [f(a), f(b)]$

```
do{
    x = uniforme(a,b)
    y = uniforme(f(a), f(b))
} while ( y > f(x) );
```

```
x = uniforme(a,b);
x = generaSin(a,b);
```

x = uniforme(0, -2);

```
double uniforme(double a, double b) {
    return a + (b-a) * (rand() / RAND_MAX);
}

double genera_sin(double a, double b) {
    double x, y;
    do {
        x = uniforme(a,b);
        y = uniforme(0,1);
    } while( y > sin(x) );
    return x;
}
```

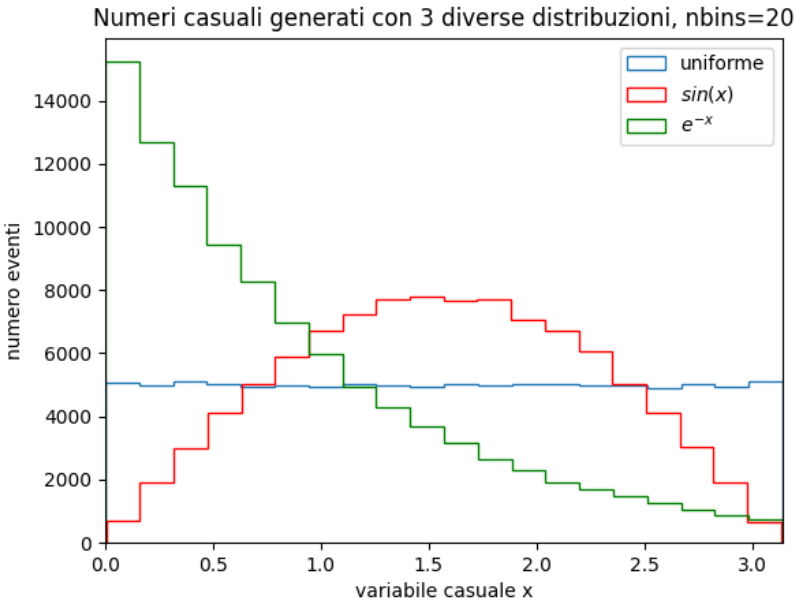



```
do {
    x = uniforme(a,b);
    y = uniforme(0,1);
} while( y > exp(-x) );
```

```c
int main() {

  srand48( time(0) );
  double a=0, b=M_PI;

  // max di sin(x) e e^(-x) per x in [0,PI]
  double ymax = 1.;

  // array per salvare i numeri generati  da utilizzare
  double piatto[NMAX];
  double sinx[NMAX];
  double expx[NMAX];
  double myfx[NMAX];

  // numero effettivo di numeri generatori per ottenere
  // NMAX numeri con la distribuzione giusta
  int npiatto=0, nsinx=0, nexpx=0, nmyfx=0;

  FILE* fp;
  char fname[] = "numeri.txt";
  fp = fopen(fname,"w+");
  if(!fp) {
    printf("errore a creare %s... exit\n", fname);
    exit(-1);
  }

  int i;
  double x,y;

  for(i=0; i<NMAX; i++) {

    if(! ((i+1)%1000) ) printf("generazione: %3d\n", i+1);
    // distribuzione uniforme
    x = uniforme(a,b);
    *(piatto+i) = x;
    npiatto++;

    do{
      nsinx++;
      x = uniforme(a,b);
      y = uniforme(0,ymax);
    } while ( y > sin(x) );
    *(sinx+i) = x;

    do{
      nexpx++;
      x = uniforme(a,b);
      y = uniforme(0,ymax);
    } while ( y > exp(-x ) );
    *(expx+i) = x;


    // scrivi output
    fprintf(fp, "%.10f\t %.10f\t %.10f\n", piatto[i], sinx[i], expx[i]);

  } // ciclo generazione
  printf("numero generazioni per piatto: %6d \t sin(x): %6d \t exp(-x): %6d\n",
   npiatto, nsinx, nexpx);


  fclose(fp);

} // main
```

```
generazione: 10000
generazione: 20000
generazione: 30000
generazione: 40000
generazione: 50000
generazione: 60000
generazione: 70000
generazione: 80000
generazione: 90000
generazione: 100000
numero generazioni per piatto: 100000    sin(x): 157295          exp(-x): 328550
```

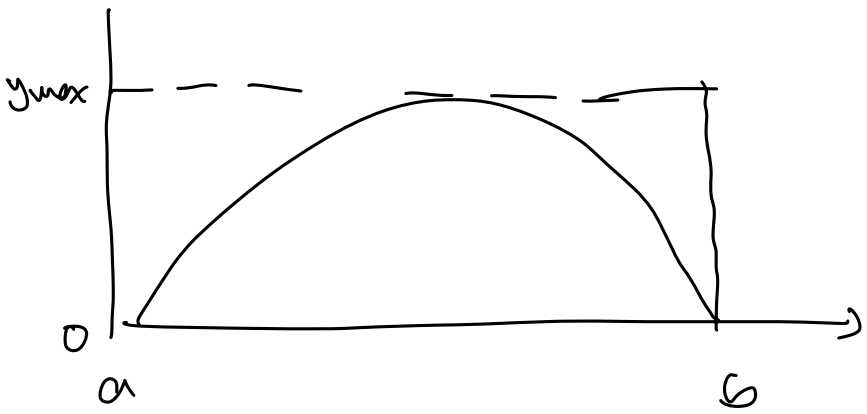
```python
import matplotlib.pyplot as plt
import numpy as np
import math as m

piatto, sinx, expx = np.loadtxt("numeri.txt", unpack=True)

nbins=20

plt.title("Numeri casuali generati con 3 diverse distribuzioni, nbins=%d"%(nbins))

plt.xlim(0, m.pi)
plt.xlabel('variabile casuale x')
plt.ylabel('numero eventi')

plt.hist(piatto, bins=nbins, histtype='step', label='uniforme')

plt.hist(sinx, bins=nbins, histtype='step', color='red', label='$sin(x)$')

plt.hist(expx, bins=nbins, histtype='step', color='green', label="$e^{-x}$")

# legenda dei 3 grafici
plt.legend(loc="upper right")


plt.show()
```



bin width = 3.14/20

python3 genera.py

• py



Area Rettangolo

$= (b-a) * y_{max}$

$$\int_a^b f(x)\,dx = \frac{N_{acc}}{N_{tot}} \text{ Area}$$