Cicli:

do/while : input di variabile entro limiti $[a,b]$

for : # iterazioni noto

while : prima verifica cond. poi esegue

Trovare zeri di una funzione

$f(x) = 0 \Rightarrow x_1, \dots, x_n : f(x_i) = 0$

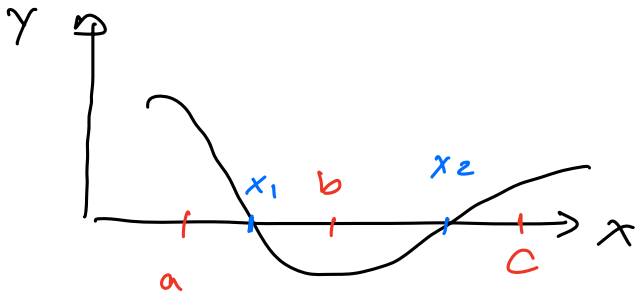
$\underbrace{3x - 4x^2 + 7 = 0}_{f(x)}$

$\sin(x) = 0$

$x^3 \sin(x) = 0$

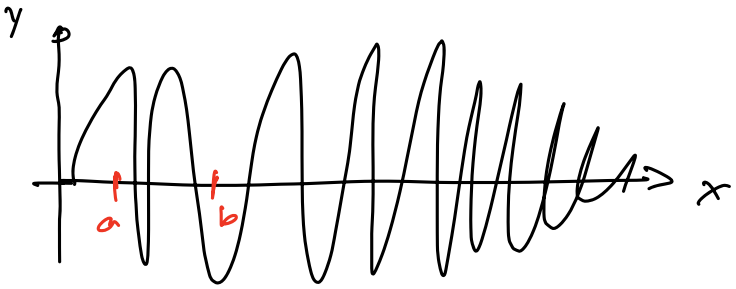
$x^2 + \tanh^2 x = 0$

Funzioni non troppo oscillanti $\Rightarrow$ Metodo di bisezione

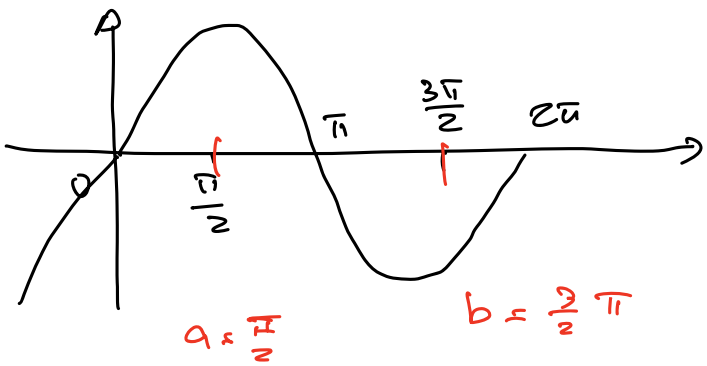


$x_1 \in [a,b]$

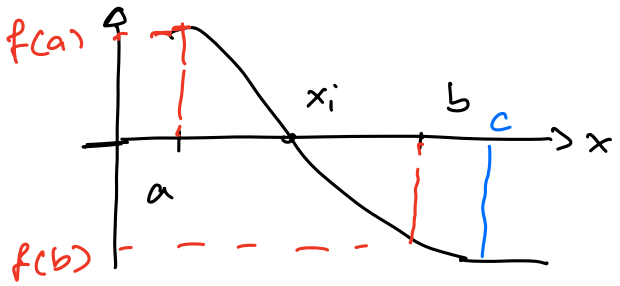
$x_2 \in [b,c]$



non funzione bene



sin(x)



$f(a) > 0$

$f(x_i) = 0$ zero della funz.

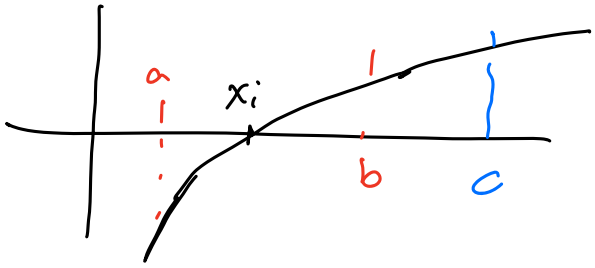
$f(b) < 0$

$f(a) \times f(b) < 0$

se $x_i \in [a,b]$

$f(b) \times f(c) > 0$

$x_i \notin [b,c]$



$f(a) < 0$

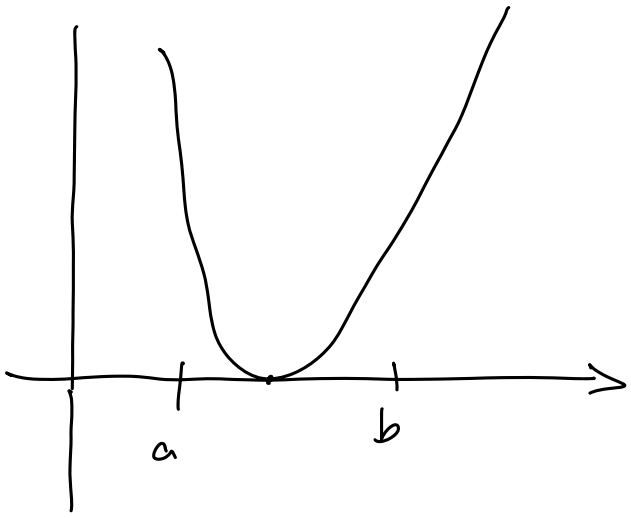
$f(x_i) = 0$

$f(b) > 0$

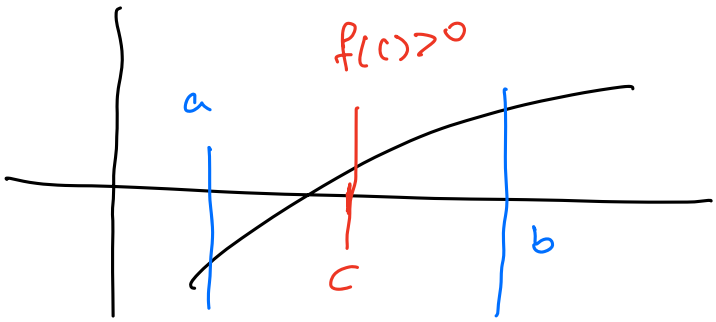
$f(a) \times f(b) < 0$

$x_i \notin [b,c]$

$f(b) \times f(c) > 0$



Richiede cura perché $f(x)$ non attraversa asse x



$f(a) < 0$

$f(b) > 0$

$f(c) \times f(b) < 0.$

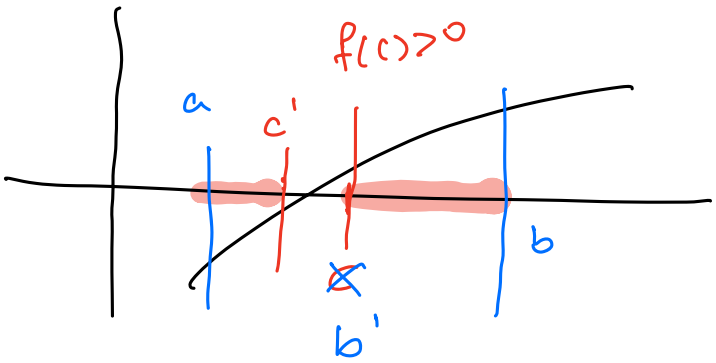
$\Rightarrow x_i \in [a,b]$

$c = \frac{a+b}{2}$ I^a iterazione.

$f(a) \times f(c) < 0$

$f(b) \times f(c) > 0 \Rightarrow$ scarto $[b,c]$

II^a iterazione



$a = a$

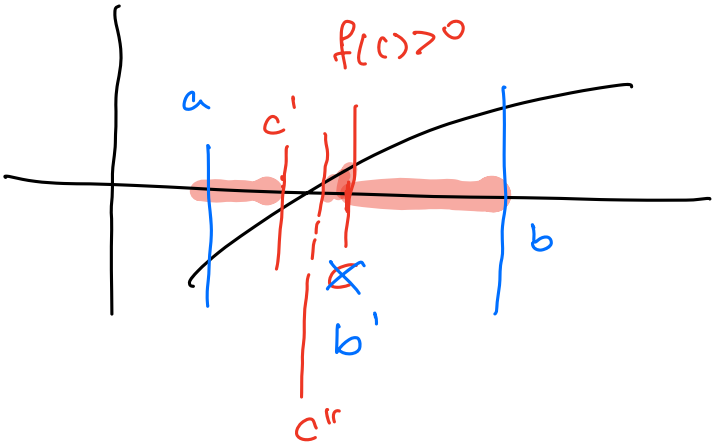
$b' = c$

$c' = \frac{a+b'}{2}$

$f(a) \times f(c') > 0$

$f(c') \times f(b') < 0 \Rightarrow x_i \in [c', b']$

III^a iterazione



$a'' = c'$

$b'' = b'$

$c'' = \frac{a'+b'}{2}$

$f(a'') \times f(c'') < 0$

$f(c'') \times f(b'') > 0$

ciclo si ferma quando raggiungo precisione prestabilita

si ferma if(fabs(a-b)) < precisione

fabs(x-y) ≡ |x-y|

$|a-b| < \varepsilon$

bisezione.c

New Open Recent Revert Save Print Undo Redo Cut Copy Paste Search Preferences Help

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

#define EPS 1.e-5

int main() {

  double a = 0., b = 0.7*M_PI, c;
  double delta = fabs(a-b);

  double p;
  int iter = 0;
  double eps = EPS;

  printf("inserisci la precisione:" );
  scanf("%lf", &eps);

  printf("calcolo zeri di cos(x)\n");

  while(delta > eps ) {
    iter++;

    c = 0.5*(a+b);
    p = cos(a)*cos(c);

    if( p > 0.) {
       a = c;
    } else if ( p < 0. ) {
       b = c;
    } else {
       a = b = c;
    }

    printf("iter: %3d  x = %.15f gradi\n", iter, c*180./M_PI);
    delta  = fabs(a-b);
  }

}
```

10^{-5}

$f(x) = \cos(x)$

$a = 0 \qquad b = 0.7 \times \pi$

$\delta = |a - b|$

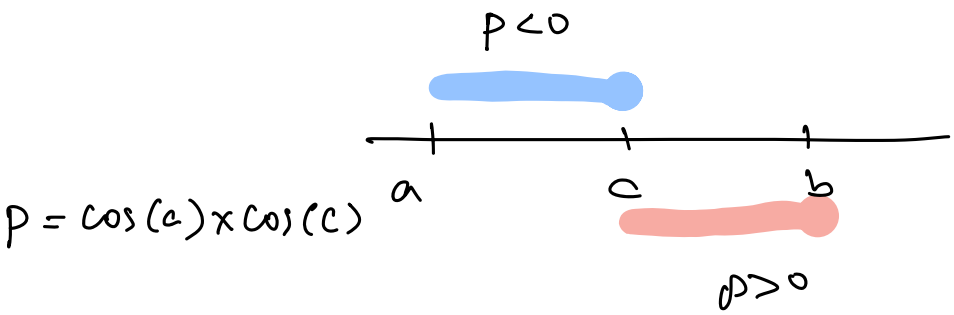
EPS: costante pre compilatore

eps: variabile double

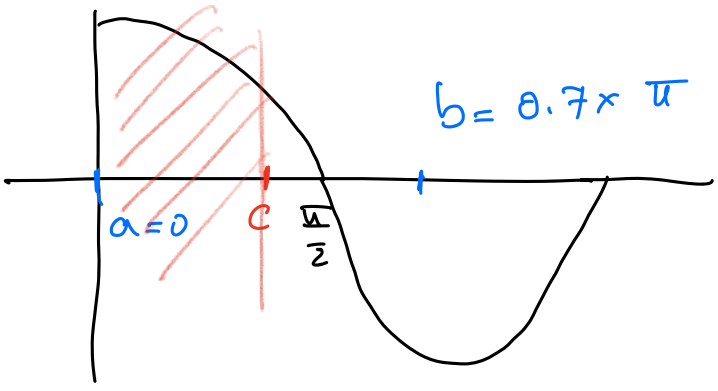
scanf("%lf", &EPS) ?

scanf("%lf", &1.e-5);

p: variabile di appoggio ausiliaria



$p = \cos(a) \times \cos(c)$



```
shamacmini:material rahatlou$ gcc -o /tmp/app bisezione.c
shamacmini:material rahatlou$ /tmp/app
inserisci la precisione:0.1
calcolo zeri di cos(x)
iter:   1  x = 63.000000000000000 gradi
iter:   2  x = 94.500000000000000 gradi
iter:   3  x = 78.750000000000000 gradi
iter:   4  x = 86.625000000000014 gradi
iter:   5  x = 90.562500000000014 gradi
shamacmini:material rahatlou$ /tmp/app
inserisci la precisione:0.001
calcolo zeri di cos(x)
iter:   1  x = 63.000000000000000 gradi
iter:   2  x = 94.500000000000000 gradi
iter:   3  x = 78.750000000000000 gradi
iter:   4  x = 86.625000000000014 gradi
iter:   5  x = 90.562500000000014 gradi
iter:   6  x = 88.593750000000000 gradi
iter:   7  x = 89.578125000000000 gradi
iter:   8  x = 90.070312500000000 gradi
iter:   9  x = 89.824218750000000 gradi
iter:  10  x = 89.947265625000000 gradi
iter:  11  x = 90.008789062500000 gradi
iter:  12  x = 89.978027343750000 gradi
shamacmini:material rahatlou$
```

90.000
89.978

$\varepsilon = 0.001$

90.0
90.6

$\varepsilon = 0.1$

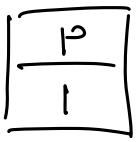
Numeri Primi

N primo ?

int n, m

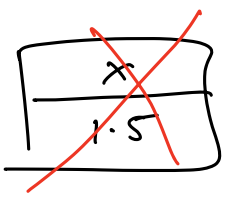
n = 3 m = 2

int p = n/m ;



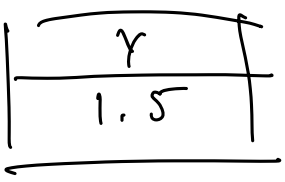
float x = n/m;

x = 1.0000



float y = (float) n / m

= 1.*n/m



int resto = n % m; resto di n/m (intero)

n = 123

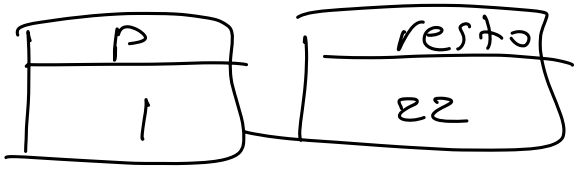
m = 295

p = n/m

resto = n % m

n = 317

m = 295



$$1\ \frac{22}{295}$$

n = 17 primo ?

n/2

n % 2 ≠ 0

n % 3 ≠ 0

n % 4 ≠ 0

n % 5 ≠ 0

n %

div_min = 2

div_max = $\sqrt{n}$ + 1

n = 15

n % 2 ≠ 0

n % 3 = 0

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

#define N 100

int main() {

  int n,j,jMax;
  printf("inserisci un numero intero: ");
  scanf("%d", &n);

  // valori tra cui cercare divisori
  j = 2;
  jMax = (int)sqrt(n)+1;

  printf("verifica fino jMax: %d\n", jMax);

  // aumenta j solo se i non divisibile per j
  for(j = 2; j < jMax; j++) {
    printf("provo con %d\n", j);
    if( n%j == 0) {
      printf("%d multiplo di %d ... non primo :(\n", n, j);
      break; // interrompe il ciclo. non molto elegante
    }
  }
  printf("j dopo ciclo: %d\n", j);
  if(j == jMax) printf("Numero primo %d\n", n);

}
```

n % j : resto di $\frac{n}{j}$

$\neq 0$

⇓

non divisibile per j

⇒ continua con j++

```
shamacmini:material rahatlou$ gcc -o /tmp/app primi1.c
shamacmini:material rahatlou$ /tmp/app
inserisci un numero intero: 16
verifica fino jMax: 5
provo con 2
16 multiplo di 2 ... non primo :(
j dopo ciclo: 2
```

```
shamacmini:material rahatlou$ /tmp/app
inserisci un numero intero: 127
verifica fino jMax: 12
provo con 2
provo con 3
provo con 4
provo con 5
provo con 6
provo con 7
provo con 8
provo con 9
provo con 10
provo con 11
j dopo ciclo: 12
Numero primo 127
```

for $j < jMax = 12$

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

#define N 100

int main() {

  int i,j,jMax;
  printf("Numeri primi tra 1 e %d:\n", N);

  // ciclo sui numeri da 1 a N
  for(i=1; i<=N; i++) {

    // valori tra cui cercare divisori
    j = 2;
    jMax = (int)sqrt(i)+1;

    // aumenta j fino jMax  solo se i non divisibile per j
    while( (j<jMax) && (i%j) ) {
      j++;
    }

    if(j == jMax) printf("%d\n", i);

  } // ciclo for

}  // main
```

$j < jMax$.

$\text{resto}\frac{i}{j} \neq 0$

resto (i/j) ≠ 0.
=> TRUE

```
shamacmini:material rahatlou$ gcc -o /tmp/app primi.c
shamacmini:material rahatlou$ /tmp/app
Numeri primi tra 1 e 100:
1
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

if (i%j != 0)

if (i%j)

≠0 ≡ TRUE

```c
if ( i%100 == 0){
    printf ("iterazione %d \n", i);
}
```

```
if ( !(i%100) ) {
    printf("iterazione %d\n", i);
}
```

Conversione base

n = 5	diviso	resto
n / 2	2	1
2 / 2	1	0
1 / 2	0	1

Ciclo {
- divisione
- resto
- stampa resto

}

	n/2	resto
n = 8	4	0
4	2	0
2	1	0
1	0	1

Ciclo {
- divisione
- resto
- stampa resto printf("%d", resto)

}

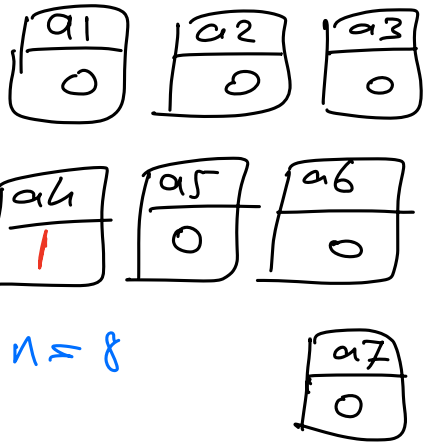
0001 Sbagliato.

1000

```
int a1, a2, a3, a4, a5, a6, a7;

printf("%d", a7);
printf("%d", a6);
   :
printf("%d", a1);
```

0001000



n = 8

⇒ array in C

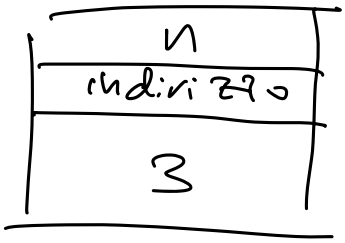
```
int a[7];
```

a array di 7 elementi di tipo intero.

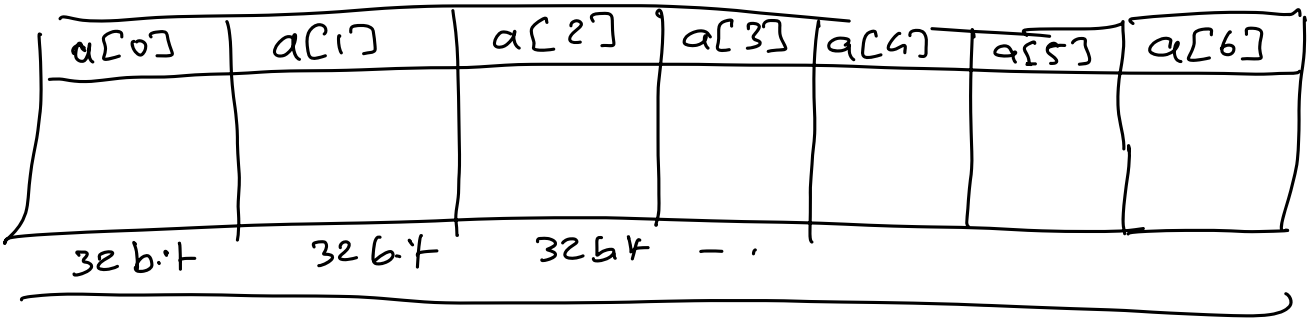
```
double x[100];
```

x array di 100 elementi di tipo double.

```
int n
```



int a[7]



size int: 32 bit

Struttura dati a[7]
grande 7 x 32 bit
= 7 x 4 byte
= 28 byte

int a[7] : a: nome di array. di tipo intero

7: lunghezza di array.
celle.

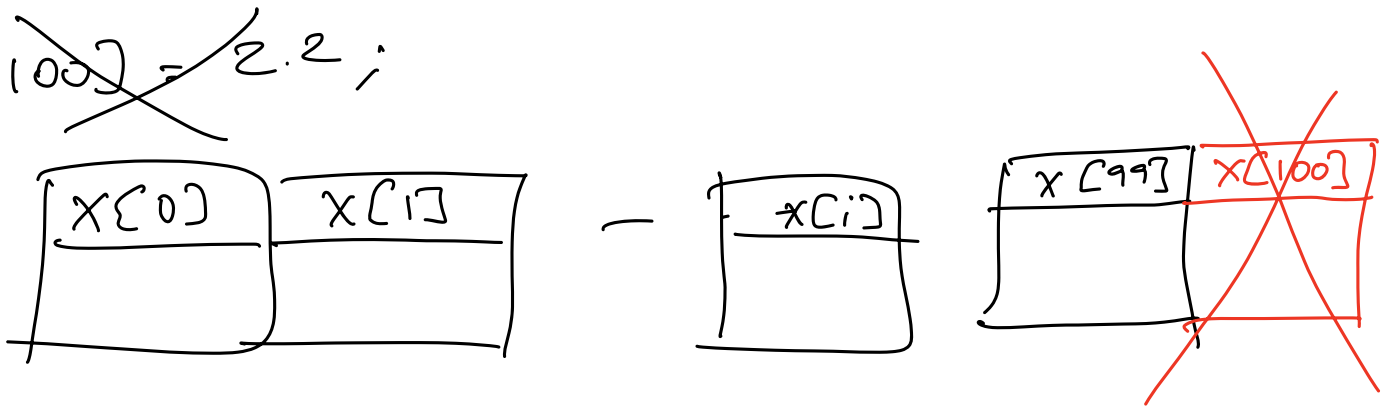
a[0], - - , a[6] : 7 celle di array.

```
a[1] = 3;
a[6] = -2;
```

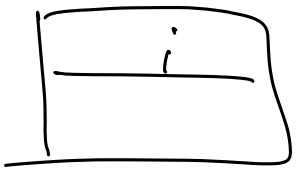
double x[100]; indice è intero.

x[98] = 0.23; indice ∈ [0, lunghezza-1]

~~x[100] = 2.2;~~




```
int i = 0;
for( i = 0; i < 100; i++) {
        x[i] =
}
```



x