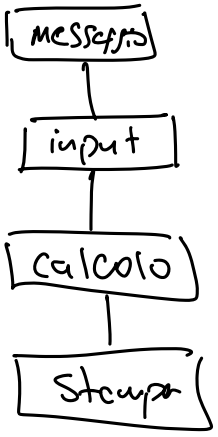
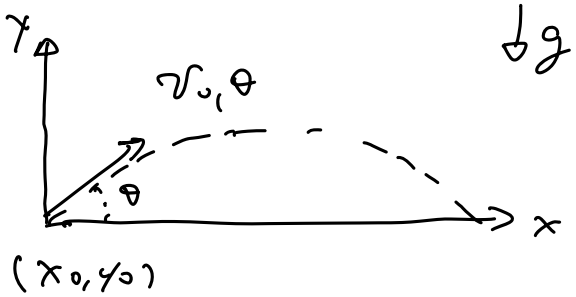
Diagramma di flusso

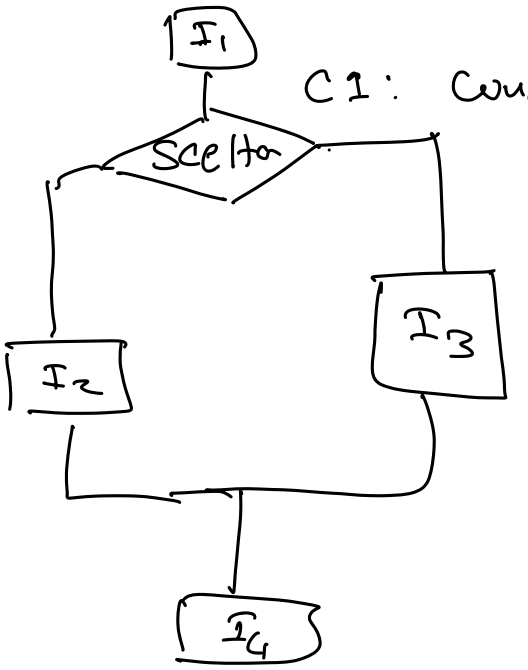


Sequenza lineare di istruzioni:



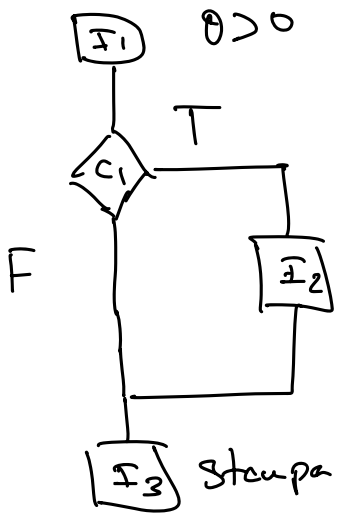
$ax^2 + bx + c = 0$

Modificare flusso

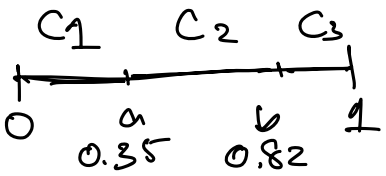


C1: Condizione

Caso semplice




```
double theta;
scanf("%lf", &theta);
if (theta > 0) {
    printf("angolo valido\n");
} else {
    printf("angolo negativo non ok\n");
}
```




```
double x;

printf("inserisci x tra 0 e 1: ");
scanf("%lf", &x);
```

C1: $0 \le x \le a$

C2: $a < x \le b$

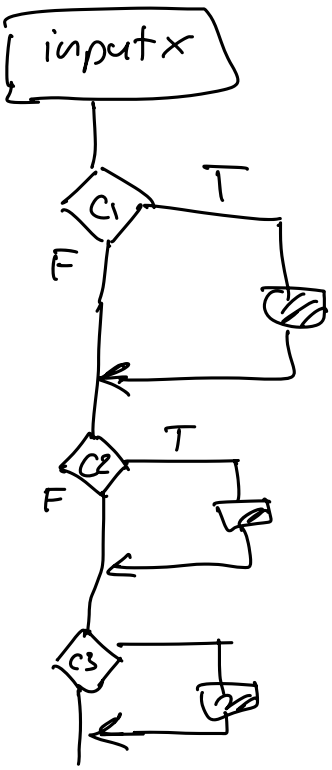
C3: $b < x \le 1$

$a = 0.35$
$b = 0.82$
$x = 0.49$

```
C1: (x >= 0) && (x <= a)
C2: (x > a) && (x <= b)
C3: (x > b) && (x <= 1)
```

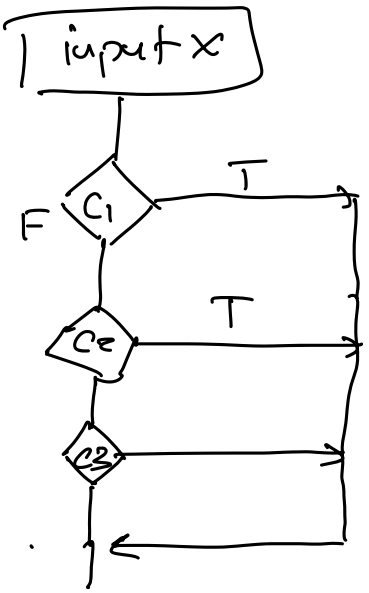
```
if ( c1 ) {
  --
}
if ( c2 ) {
  --
}
if (c3) {
  ==
}
```

```
if ( c1 ) {   F
  ==
}
if ( c2 ) {   T
  ==
}
if ( c3 ) {   F
  ==
}
```



Implementazione non esclusive delle condizioni

```
if(c1) {
   ==
} else if(c2) {
   ≡
} else if(c3) {
   ≡
} else {
   printf("x inserito non validoɴ");
}
```




```
if((x>=0)&&(x<=a)) {
   ≡
} else if(x<=b) {
   =
} else if(x<=1) {
   ≡
} else {
   ≡
}
```

) if / else if / else

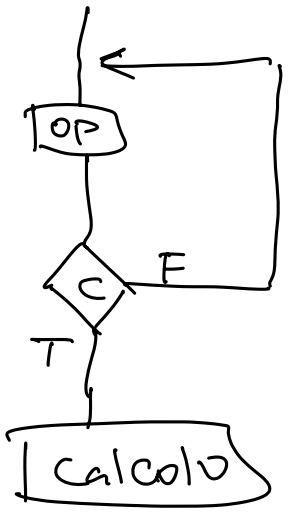
Condizioni / Decisioni

Iterazioni

input x

Richiedi x > 0

calcolo



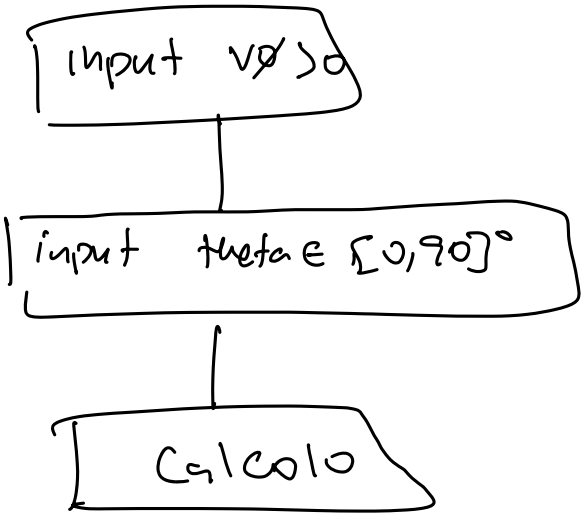
Iterazione / Ciclo

C: $x > 0$ && $x <= 1$

do / while

```
do {
    printf("inserisci 0<x<1: ");
    scanf("%lf", &x);
} while ((x<0) || (x>1));
```

$!((x >= 0) \&\& (x <= 1))$




```
do {
    ≡≡≡
} while (7);
```

Ciclo infinito

```
do {
    ≡≡≡
} while (condiz);
```

```
double x;
do {
    printf("Inserisci x: ");
    scanf("%lf", &x);
} while (x = 0);
```

VERA `x = 0`: assegna 0 alla variabile x

dipende `x == 0`: controlla se x uguale a 0

Consiglio: usare `==` tra interi

libreria matematica usare `>=, >, <, <=` con float/double.

`fabs(x) <= epsilon` $|x| < \varepsilon$

↓

tolleranza, precisione.

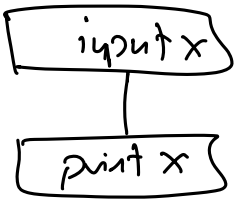
`fabs(x-l) <= epsilon` ✓

`(x-l) == 0` ✗

ciclo do-while

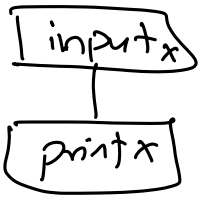
```
do {
} while (x<0);
```

ottiene: $x \geq 0$.




```
do {
} while (x>=0);
```

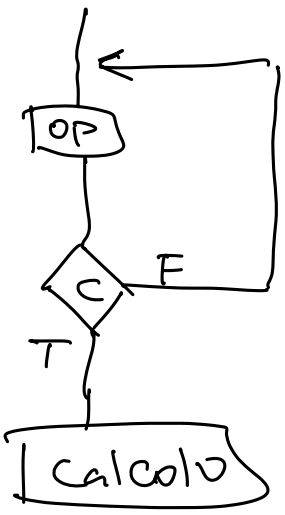
ottiene: $x < 0$



apt-get install emacs

Ciclo while-(do)

```
while ( condiz ) {
   ≡
}
```




```
float totale = 100.;
while ( totale > 0 ) {
   ≡ stupidino
   // estrai a caso spesa.
   printf(" quanto spendi? ");
   scanf(" %.1f", &spesa );
   totale -= spesa;
}
```

ciclo while

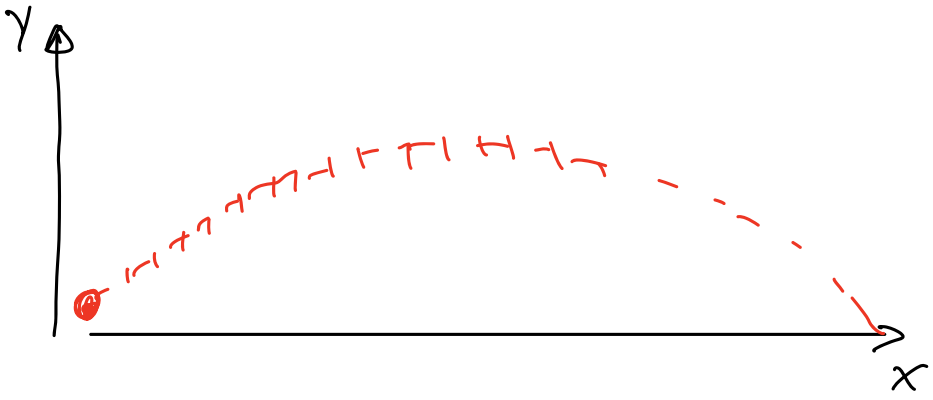
totale -= spesa; ⟷ totale = totale - spesa

do-while

```
do {
   scanf("%d", &n);
} while (n<10);
```

while

```
scanf("%d", &n);
while (n<10) {
   scanf("%d", &n);
}
```




```
int n=0;
x0 = 0;
y0 = 0;
do {
   calcolo x
   calcolo y
   calcolo vx
   calcolo vy
   n++;
} while(y>0);
printf("%d passi", n);
```

```
do {
    int n = 0;
    Calcolo numeros:
    =
    =
    =
    n++;
} while ( cond eseguite );
```

Calcolo y max:

```
double ymax = 0; double xmax = 0;   } variabili di appoggio ausiliarie.
do {
    Calcolo x
    Calcolo y
    Calcolo vx
    Calcolo vy
    if ( y > ymax ) {
        ymax = y;
        xmax = x;
    }
} while ( y > 0 );
```

Fattoriale: $n! = n \times n-1 \times n-2 \times \dots \times 1$

unsigned int — 32 bit senza segno

$2^{32}-1$ numeri

$max = 2^{32}-1$

unsigned long long int p; — 64 bit senza segno

$2^{64}-1$ come valore max.

```
int i;
printf("n = ");
scanf("%d", &n);
printf("Calcolo del fattoriale n! \n");
for ( i=1 ; i<=n ; i++ ) {
    printf("i = %d \n", i);
}
```

inizio iteratore

fine iterazione

incremento iteratore

i = 1

i = 2

i = 3

⋮

i = n

```
for (i=0; i<n; i++){
    printf("i = %d \n", i);
}
```

i = 0

i = 1

⋮

i = n-1

```
for( i = -10; i < 50; i += 5 ) {
    printf( - - )
}
```

i = -10
i = -5
i = 0

i = 5

i = 45

```
for( i = 90; i > 12; i - - ) {

}
```

i = 90
i = 89

i = 13