

Ciclo do/while

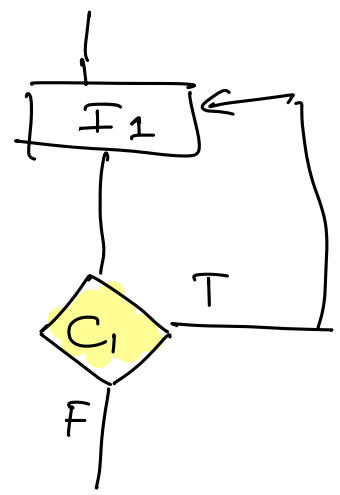
do {

=====

istruzione I_1

} while (condizione C_1);

Acquisizione
solo do/while



Ciclo while ?

oggetto libero $v_0 = \emptyset$

$$y(t) = h - \frac{1}{2} g t^2$$

double $h = 10$; // metri:

double $y = h$;

double $t = \emptyset$;

double $dt = 0.1$; // Sec

while ($y > 0$) {

$t = t + dt$; // equivale a $t = t + dt$

$$y = h - 0.5 * 9.8 * t * t;$$

printf("t: %.2f y: %.2f \n", t, y);

}

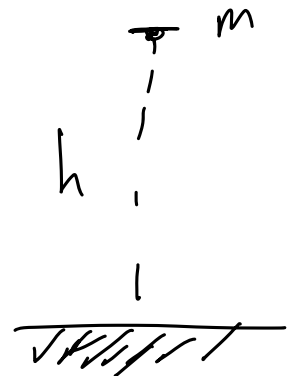
Iniz: 0.

$$y = 10 - 0.049$$

$$9.951$$

t: 0.10 y: 9.95

$$y(t=0) = h$$



$$g = 9.8 \frac{m}{s^2}$$

$$y(t) = h - v_0 t - \frac{1}{2} g t^2$$

dt
0.1

y
10

t
0

dt
0.1

y
9.951

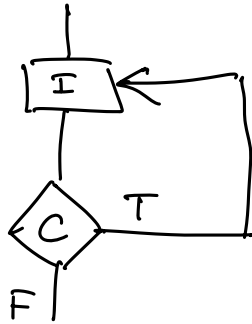
t
0.1

9.95

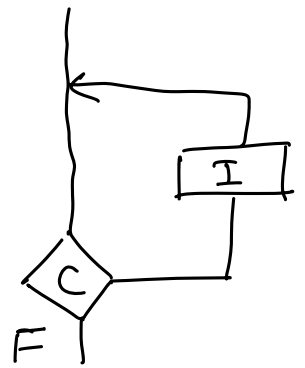
0.2

Diagramme di flusso ciclo while:

do/while



ciclo while



double $G = 9.81;$

while ($y > 0$) {

$t += dt;$ // equivale a $t = t + dt$

$y = h - 0.5 * G * t * t;$

printf("t: %.2f y: %.2f \n", t, y);

}

G: una costante non variabile

~~#define~~ GRAV 9.81

int main() {

Istruzione
pre-compilazione
per la costante
GRAV

≡

$y = h - 0.5 * GRAV * t * t;$

~~GRAV = 9.1;~~ non compila.

#define GRAV Scherzetto

```

/* Some useful constants. */
#ifdef __USE_MISC || defined __USE_XOPEN
# define M_E      2.7182818284590452354 /* e */
# define M_LOG2E  1.4426950408889634074 /* log_2 e */
# define M_LOG10E 0.43429448190325182765 /* log_10 e */
# define M_LN2    0.69314718055994530942 /* log_e 2 */
# define M_LN10   2.30258509299404568402 /* log_e 10 */
# define M_PI     3.14159265358979323846 /* pi */
# define M_PI_2   1.57079632679489661923 /* pi/2 */
# define M_PI_4   0.78539816339744830962 /* pi/4 */
# define M_1_PI   0.31830988618379067154 /* 1/pi */
# define M_2_PI   0.63661977236758134308 /* 2/pi */
# define M_2_SQRTPI 1.12837916709551257390 /* 2/sqrt(pi) */
# define M_SQRT2  1.41421356237309504880 /* sqrt(2) */
# define M_SQRT1_2 0.70710678118654752440 /* 1/sqrt(2) */
#endif

```

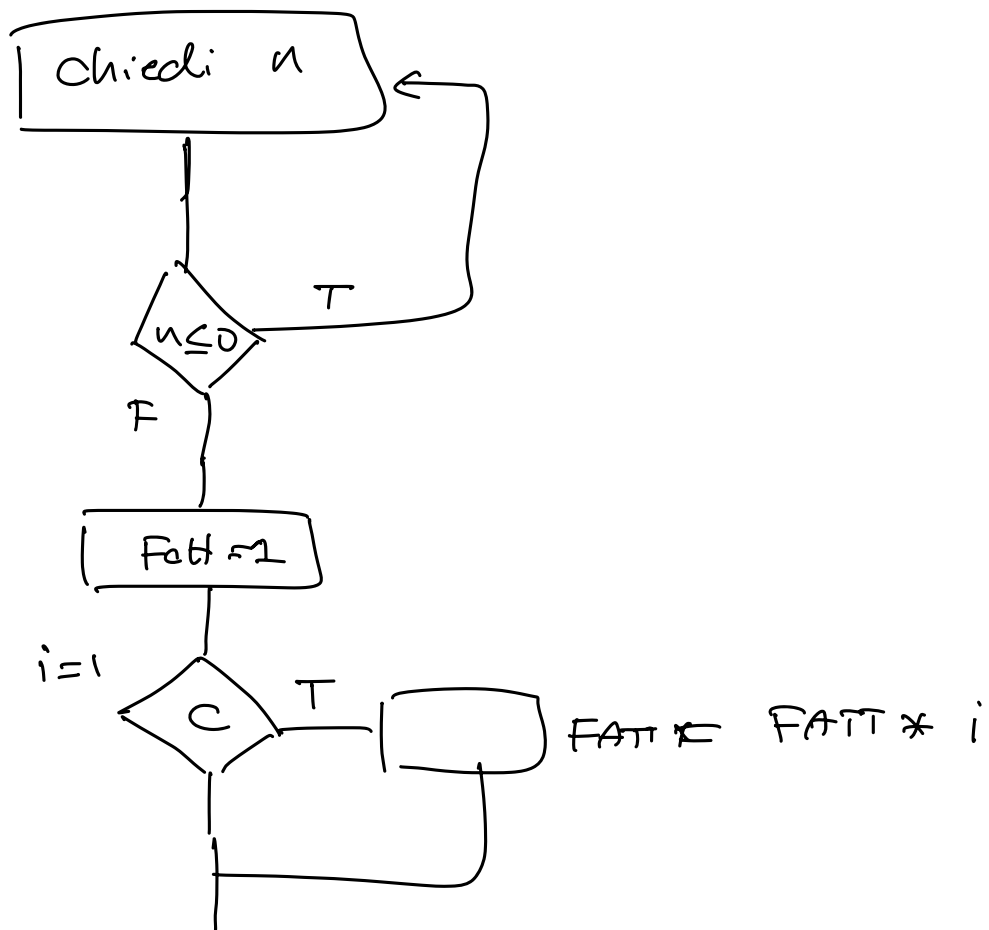
Ciclo FOR: ciclo con #iteration: noto

$$n! = n \cdot n-1 \cdot n-2 \cdot n-3 \cdot \sim \cdot 2 \cdot 1$$

for(cond iniz; ; cond fin ; modifica iteratore) {

≡

}



```
#include <stdio.h>

int main() {
    int n, i;
    // 64 bit senza segno => max = 2^64 - 1 = 18.5 x 10^18
    unsigned long long int fattoriale = 1;

    do{
        printf("Inserisci numero intero positivo: ");
        scanf("%d", &n);
    } while( n<=0 );    ! (n>0)

    for (i = 1; i <= n; i++) {
        // moltiplicazione iterativa
        fattoriale *= i;
    }

    printf("fattoriale di %d = %llu\n", n, fattoriale);
}
```

$$n = 5$$

i è iteratore

fattoriale = fattoriale * i;

i	fattoriale
✓ 1	1
✓ 2	2
✓ 3	6
✓ 4	24
✓ i = n 5	120
X i <= n 6	<u> </u>

```
shamacmini:material rahatlou$ gcc -o /tmp/app fattoriale.c
shamacmini:material rahatlou$ ./tmp/app
Inserisci numero intero positivo: 56
fattoriale di 56 = 6908521828386340864
shamacmini:material rahatlou$
```

Successione ricorsive

$$a_{n+1} = a_n + 2.$$

$$a_n = \frac{1}{n}$$

Fibonacci 0 0 1 1 2 3 5

$$a_{n+2} = a_{n+1} + a_n$$

$$a_n = a_{n-1} + a_{n-2}$$

```

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

int main() {

    unsigned long long int a0, a1, an;
    int i, n;

    a0=0;
    a1=1;

    printf("calcolo della serie di Fibonacci\n");

    do{
        printf("Quanti elementi della serie vuoi calcolare (0 < n < 100)? ");
        scanf("%d", &n);
    } while( n<0 || n>=100 );

    printf("a_1 : %llu\n", a1);

    for(i=2; i<n+1; i++) {

        an = a0 + a1;
        printf("a_%-2d: %llu\n", i, an);

        a0 = a1;
        a1 = an;

    }

}

```

! (0 < n < 100)

a_0, a_1 : variabili
ausiliare
o di appoggio

i	a_n
2	1
3	2
4	3

```

shamacmini:material rahatlou$ gcc -o /tmp/app fiboLLU.c
shamacmini:material rahatlou$ /tmp/app
calcolo della serie di Fibonacci
Quanti elementi della serie vuoi calcolare (0 < n < 100)? 13
a_1 : 1
a_2 : 1
a_3 : 2
a_4 : 3
a_5 : 5
a_6 : 8
a_7 : 13
a_8 : 21
a_9 : 34
a_10: 55
a_11: 89
a_12: 144
a_13: 233
shamacmini:material rahatlou$ █

```