



Cicli

Recap

```
int num1 = -2;  
int num2 = 4;
```

Il costrutto `if` permette di eseguire blocchi di codice condizionatamente al valore di verità di una condizione.

```
if(num1 > 0)  
{  
    //This is not executed  
}  
else if(num2 > num1)  
{  
    //This is executed  
}  
else  
{  
    //This is not executed  
}
```

In C, *vero* significa diverso da 0.

In una catena di `if`, `else if`, `else`, solo un ramo viene eseguito, anche se più condizioni sono vere.

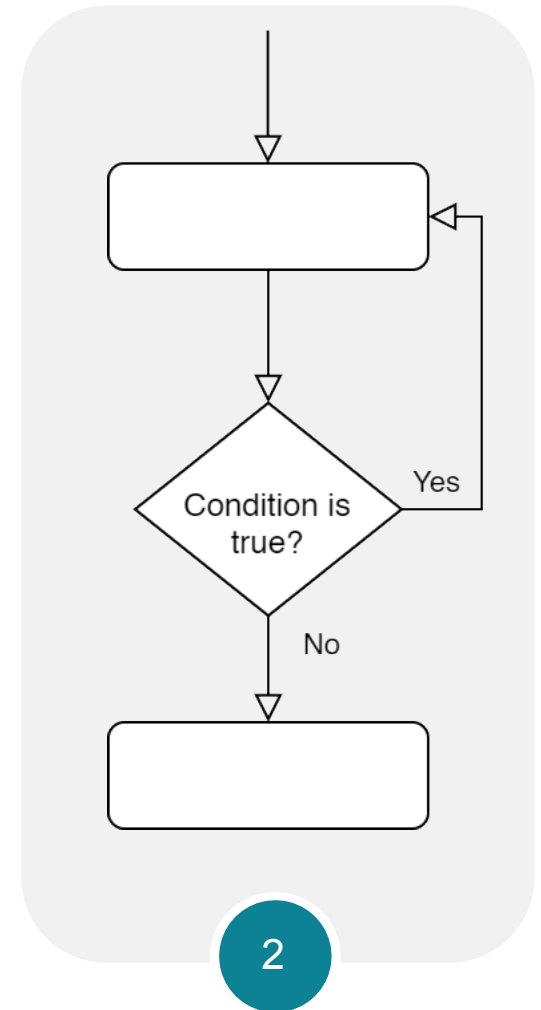
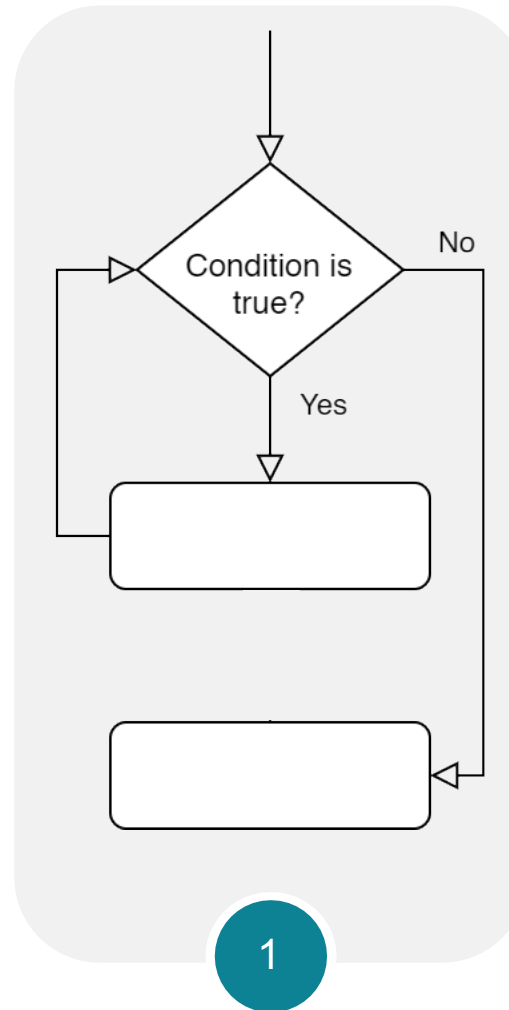
Ripetere blocchi di istruzioni

Cicli

Molte volte è richiesto ripetere le stesse operazioni (o gruppo di operazioni) per più volte.

I *cicli* ci permettono di ripetere tutte le istruzioni all'interno di un blocco di codice.

- 1 Nei cicli di tipo *while* la condizione viene controllata prima di eseguire le istruzioni del blocco.
Le istruzioni potrebbero non essere mai eseguite
- 2 Nei cicli di tipo *do-while* la condizione viene controllata dopo aver eseguito le istruzioni del blocco.
Le istruzioni vengono quindi eseguite almeno una volta



Cicli di tipo while (1)

Ciclo while

Il ciclo *while* permette di eseguire un blocco di istruzioni finché una determinata condizione è vera.

Uscire dal ciclo

⚠ Se all'interno del ciclo non eseguiamo istruzioni che influenzano la condizione, potremmo rimanere nel ciclo per sempre.

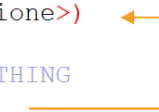
```
while (<condizione>)  
{  
    //DO SOMETHING  
}
```

i Alcune parole chiave ci permettono di influenzare il flusso di esecuzione all'interno dei cicli.


break termina l'esecuzione del ciclo istantaneamente e salta all'istruzione dopo il ciclo.

continue termina l'iterazione corrente del ciclo e salta alla prima istruzione del ciclo (che per i cicli di tipo *while* è il controllo della condizione).

```
while (<condizione>)  
{  
    //DO SOMETHING  
    continue;  
    //DO SOMETHING  
}  
//DO SOMETHING
```



```
while (<condizione>)  
{  
    //DO SOMETHING  
    break;  
    //DO SOMETHING  
}  
//DO SOMETHING
```



Esercizio 1

ES1: Scrivere un programma che chiede all'utente un numero intero e che stampa tutti i numeri positivi minori o uguali al numero dato.

Esercizio 1 - Soluzione

```
void main()  
{  
    int i;  
    printf("Inserisci il numero:");  
    scanf("%d", &i);  
  
    while(i>0)  
    {  
        printf("%d\n", i);  
        i--;  
    }  
}
```

Esercizio 2

ES2: Cosa succede a questo programma?

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

void main()
{
    int i=1;
    while(i>0)
    {
        i++;
    }
    printf("%d", i);
}
```

Esercizio 2 - Soluzione

ES2: Cosa succede a questo programma?

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

```
void main()
{
    int i=1;
    while(i>0)
    {
        i++;
    }
    printf("%d", i);
}
```

Dopo qualche secondo, il programma stampa questo numero:

-2147483648

Vi dice qualcosa?

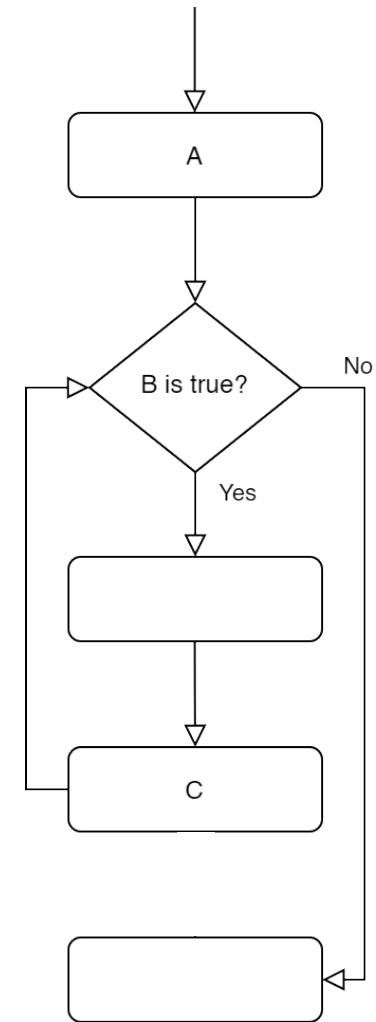
Cicli di tipo while (2)

Ciclo for

Il ciclo for è un ciclo di tipo *while*, quindi la condizione viene verificata prima di eseguire le istruzioni del blocco.

In più, il ciclo for permette di eseguire una istruzione all'ingresso nel ciclo e una al termine di ogni iterazione.

```
int i;  
for (A; B; C)  
{  
    //DO SOMETHING  
}
```



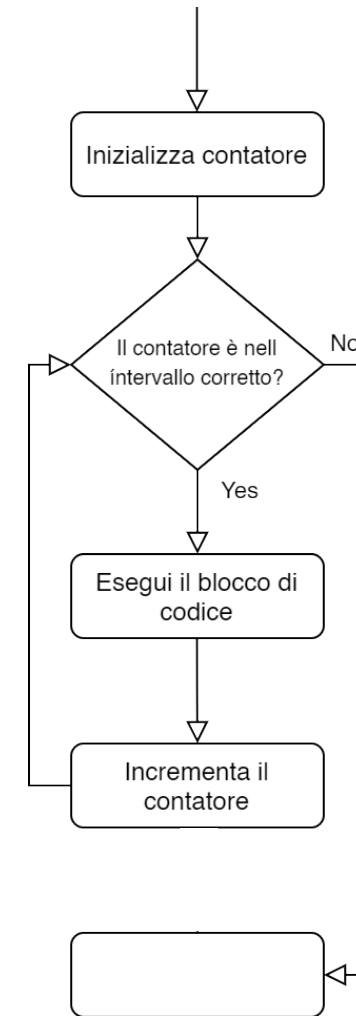
Cicli di tipo while (3)

Ciclo for

Il ciclo for è molto utile quando si deve eseguire del codice per un numero di volte specifico o quando il numero di iterazioni è determinato dal valore di una variabile.

In questi casi, possiamo usare un *contatore* che viene incrementato ad ogni iterazione.

```
int i;  
for (i=0; i<10; i++)  
{  
    //DO SOMETHING  
}
```



Esercizio 3

ES3: Scrivere un programma che chiede all'utente un numero intero e che stampa la somma di tutti i numeri interi positivi minori o uguali al numero dato.

Esercizio 3 - Soluzione

```
int num;

printf("Inserisci il numero:");
scanf("%d", &num);

int i, res = 0;

for(i=1; i<=num; i++)
{
    res += i;
}
printf("%d", res);
```

Esercizio 4

ES4: Scrivere un programma che stampa a schermo un triangolo rettangolo composto da asterischi, di altezza definita dall'utente mediante un numero intero

Es: se l'altezza è 7, stampa:

```
*  
**  
***  
****  
*****  
*****  
*****
```

SUGGERIMENTO: dovete usare due cicli for annidati

Esercizio 4 - Soluzione

```
int h;

printf("Inserisci l'altezza:");
scanf("%d", &h);

int riga, colonna;

for(riga=1; riga<=h; riga++)
{
    for(colonna=0;colonna<riga;colonna++)
    {
        printf("*");
    }
    printf("\n");
}
```

Esercizio 4 - Extra

ES4: Estendere il programma dell'es. 4 per poter disegnare anche un rettangolo ed il perimetro di un rettangolo. Permettere di scegliere all'utente la forma desiderata.

* *

* *

*

**

Esercizio 5

ES5: Scrivere un programma che chiede in input un numero e stampa tutte le potenze di 2 minori o uguali del numero dato.

CHALLENGE: fare due versioni, una con un ciclo for e una con un ciclo while.

Esercizio 5 – Soluzione (for)

```
int a;  
int c;  
printf("Enter number: ");  
scanf("%d", &a);  
  
for(c=1; c<=a; c*=2)  
{  
    printf("%d\n", c);  
}
```

Esercizio 5 – Soluzione (while)

```
int a;  
int c=1;  
printf("Enter number: ");  
scanf("%d", &a);
```

Cosa succede se non inizializzo?

```
while(c<=a)  
{  
    printf("%d\n", c);  
    c = c*2;  
}
```

Cicli di tipo do-while (1)

Ciclo do-while

Il ciclo *do-while* permette di eseguire un blocco di istruzioni finché una determinata condizione è vera.

Poiché la condizione è controllata al termine delle iterazioni, il codice è eseguito sempre almeno una volta.

⚠ Attenzione al punto e virgola al termine dell'istruzione while.

```
do
{
    //DO SOMETHING
}
while (<condizione>);
```

Cicli di tipo do-while (2)

Ciclo do-while

Il ciclo *do-while* è utile quando vogliamo eseguire codice finché non otteniamo il risultato desiderato.

Per esempio, viene usato molto per chiedere all'utente di reinserire un dato quando questo non rispetta alcune specifiche.

```
int i;  
do  
{  
    printf("Numero tra 0 e 10:");  
    scanf("%d", &i);  
}  
while(i<0 || i>10);
```

Esercizio 6

ES6: Scrivere un programma che chiede all'utente un numero e che accetta solo un numero pari

Esercizio 6 - Soluzione

```
int i;  
do  
{  
    printf("Numero pari:");  
    scanf("%d", &i);  
}  
while(i%2);
```

Esercizi aggiuntivi

Esercizio 7

ES7: Scrivere un programma che chiede all'utente due numeri n , k e che calcola n^k .

Esercizio 8

ES8: Scrivere un programma che chiede all'utente un numero e che stabilisce se questo è primo.