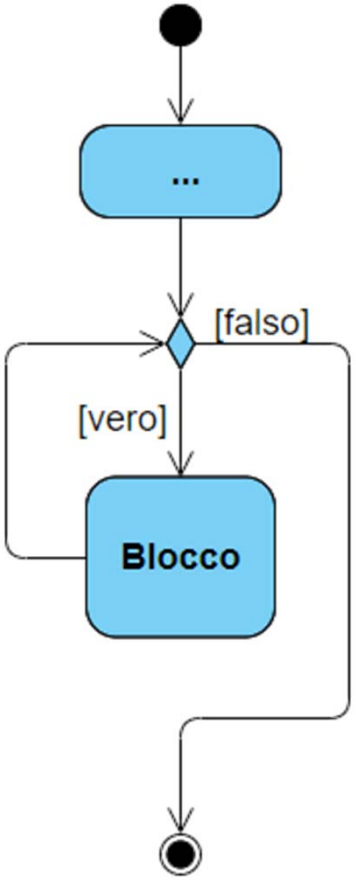
IL CICLO WHILE

- Il costrutto "*fintanto che predicato* allora fai istruzioni" si mappa in C++ come segue:

```
while (predicato) {
   //Blocco istruzioni
}
```

- La parola chiave **while** è seguita da un predicato fra parentesi
- Il predicato viene valutao e, se vero, viene eseguito il blocco di istruzioni che segue la parentesi chiusa
- Alla fine dell'esecuzione del blocco, il predicato viene valutato ancora e se vero, il blocco viene eseguito nuovamente



- **Il blocco di istruzioni viene eseguito ripetutamente finché il predicato non diviene falso**

ESEMPIO CICLO WHILE

- La variabile *var* è **inizializzata** a 0. Il **while** si traduce in:
  - "Fintanto che *var* è minore di dieci, esegui il blocco di codice tra parentesi"
- La variabile è incrementata di uno ad ogni ciclo
- Dunque, dopo 10 iterazioni il while terminerà

```
#include <iostream>
using namespace std;
int main() {
      int var=0;
      while (var<10){
          cout<<"Dentro ciclo While: "<< var+1 <<endl;
          var++;
      }
}
```

Output

```
> Dentro ciclo While: 1
> Dentro ciclo While: 2
> Dentro ciclo While: 3
> Dentro ciclo While: 4
> Dentro ciclo While: 5
> Dentro ciclo While: 6
> Dentro ciclo While: 7
> Dentro ciclo While: 8
> Dentro ciclo While: 9
> Dentro ciclo While: 10
```

CICLI INFINITI

- Può succedere che per un errore nel codice, oppure per una scelta progettuale si generi un **ciclo infinito**.
- Significa che il predicato non cambia, è sempre vero
- **NOTA -** Se dovesse succedere, per terminare l'esecuzione sul terminale è necessario digitare insieme due tasti: **ctrl** e **c**

```
int var=0;
while (true){
        var=var+1;
    }
}
```

```
int var1=0;
int var2=5;
while (var2!=0){
        var1=var1+1;
    }
}
```

In ambo i casi la variabile è incrementata all'infinito

CICLO DO-WHILE

- Il ciclo **do-while** è una particolare forma di while
- La differenza fondamentale tra il ciclo **do-while** e il ciclo **while** è che il **do-while** esegue l'esecuzione del ciclo almeno per una volta.

```
int main() {
     int var=0;
     do{
         cout<<"Dentro ciclo While"<<endl;
         var--;
     }while (var>0);
}
```

```
int main() {
     int var=0;
     while (var>0){
         cout<<"Dentro ciclo While"<<endl;
         var--;
     }
}
```

- Nel primo caso il predicato è valutato dopo aver eseguito le istruzioni. Nel secondo caso il predicato è valutato prima di eseguire le istruzioni

CICLO FOR – 1/2

- Il **for** è un altro ciclo iterativo che può essere sempre ricondotto ad un **while**.
  - A differenza del while è apriori noto il numero di iterazioni del ciclo
- Ha la seguente sintassi:

```
for(valore_iniziale; condizione_di_test; incremento)
{
   (<istruzioni da eseguire all'interno del ciclo>)
}
```

- Il valore iniziale indica quale sarà la **variabile contatore** nel ciclo e ne impostiamo il valore iniziale
- La **condizione di test** rappresenta la condizione da verificare ad ogni passo del ciclo per valutare se è necessario continuare oppure uscire dall'iterazione
- L'**incremento** descrive descrive come modificare, incrementare o decrementare il contatore ad ogni esecuzione

CICLO FOR – 2/2


```
int totale=0;
for(int i=0;i<=10; i++){
    totale=totale+i;
}
cout<<"Il totale è:"<<totale;
```

- Definiamo il **valore iniziale** attraverso l'espressione int *i*=0, con cui oltre a indicare la variabile i come variabile contatore inizializzata a 0, ne effettuiamo anche la dichiarazione.
- Nella **condizione di test** verifichiamo che *i* sia minore o uguale a 10, per rimanere all'interno del ciclo.
- Nell'espressione **incremento** ci limitiamo ad aggiungere 1 alla variabile *i* ad ogni passo del ciclo
- Il valore finale di i sarà 11 e non 10 perchè il conteggio è iniziato da 0 ed essendoci la condizione <= (invece di <) il codice è eseguito 11 volte

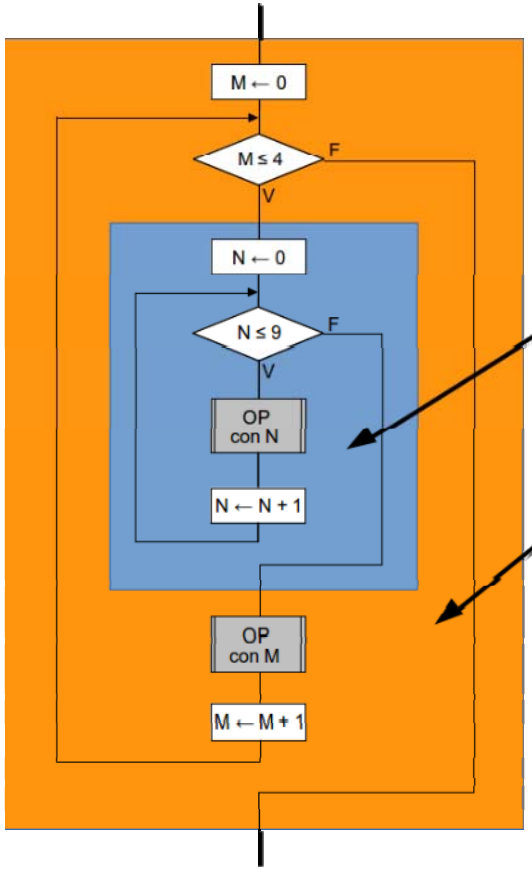
EQUIVALENZA FOR-WHILE


```
int i=0;
int totale=0;
while (i<=10){
    totale=totale+i;
    i++;
}
```

```
int totale=0;
for(int i=0;i<=10; i++){
    totale=totale+i;
}
cout<<"Il totale è:"<<totale;
```

COSTRUTTI ANNIDATI – 1/2

- E' chiaramente possibile definire costrutti, di tipo ciclico e/o selettivo, **annidati**. Ad esempio, nel caso del **for**:



Ciclo interno

Ciclo esterno

```
for(M = 0; M <= 4; M++)
{
    for(N = 0; N <= 9; N++)
    {
        // blocco OP con N
    }

    // blocco OP con M
}
```

ESEMPIO FOR


```
#include<iostream>
#include<string>

using namespace std;

/* main */
int main() {
   cout<<"Quanti elementi vuoi inserire?"<<endl;
   int num_elems;
   cin>>num_elems;
   int num,sum;
   sum=0;
   for (int i=0;i<num_elems;i++){
     cout<<"Fornisci un numero: ";
     cin>>num;
     sum = sum + num;
   }
   cout<<"La somma e': "<<sum<<endl;
   return 0;
}
```

```
osboxes@osboxes:~/Documents/examples$ ./for_ex
Quanti elementi vuoi inserire?
3
Fornisci un numero: 3
Fornisci un numero: 4
Fornisci un numero: 3
La somma e': 10
osboxes@osboxes:~/Documents/examples$
```

ESEMPIO MENU – SWITCH E DO-WHILE


```
#include<iostream>

using namespace std;

/* main */
int main() {
    int answer;

    cout << "Pick a choice from the list: " << endl;
    cout << "Pick from choices 1 (run), 2 (walk), or 3 (talk), depending on what you want from
the menu"<<endl;

    do{
        cin >> answer;
        switch (answer){
            case 1:
                cout << "I want to run"<<endl;
                break;

            case 2:
                cout << "I want to walk"<<endl;
                break;
            case 3:
                cout << "I just want to talk to my friends."<<endl;
                break;
            default:
                cout << "Bad choice! Please try again"<<endl;
        }
    } while (answer <= 0 || answer > 3);

    return 0;
}
```

```
osboxes@osboxes:~/Documents/examples$ ./switch_do__ex
Pick a choice from the list:
Pick from choices 1 (run), 2 (walk), or 3 (talk), depending on what you want fr
m the menu
5
Bad choice! Please try again
4
Bad choice! Please try again
2
I want to walk
osboxes@osboxes:~/Documents/examples$
```

ESERCIZI

1. Scrivere un programma che richiede 10 numeri decimali all'utente per farne poi la media e stamparne il risultato a video. Il programma dovrà anche verificare che l'utente non inserisca valori minori di zero. In tal caso, il numero non dovrà essere considerato nel calcolo della media.
   - Il programma dovrà essere scritto sia usando il **for** che il **while**

ESERCIZI DI AUTOVALUTAZIONE

1. Scrivere un programma che somma i primi n interi (n fornito dall'utente)
2. Scrivere un programma che somma i numeri divisibili per 4 e 7 compresi tra 0 e 100
3. Scrivere un programma che somma i primi 10 numeri divisibili sia per 5 che per 8
4. Scrivere un programma che conta da 1 a 100 sommando i numeri pari e sottraendo i numeri divisibili per 3 (se un numero è pari e divisibile per 3 va prima sommato e poi sottratto, ossia saltato)
5. Scrivere un programma che dato un numero assegnato (tra 0 e 1000), chiede all'utente di indovinare il numero.
   - Ad ogni tentativo errato il programma suggerisce all'utente se il numero da indovinare è > o < del numero di tentativo.
   - L'utente deve indovinare in 10 tentativi altrimenti perde
6. Considerando lo stesso gioco del punto precedente, invertire i ruoli: l'utente pensa il numero e il sistema prova ad indovinare, ad ogni tentativo proposto dal programma, l'utente deve rispondere con una tra tre opzioni: indovinato, minore, maggiore. Il programma andrà avanti finché non indovina il numero.

ULTERIORI ESERCIZI

HackerRank:

- https://www.hackerrank.com/challenges/c-tutorial-for-loop/problem