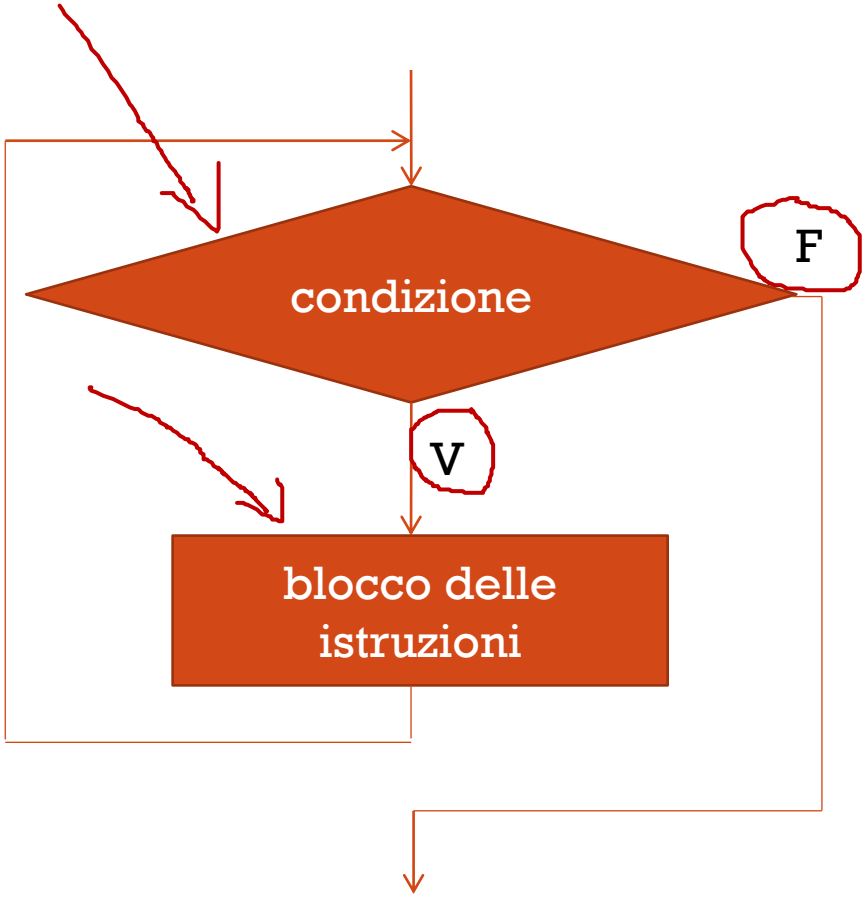
IL COSTRUTTO ITERATIVO

Il linguaggio C

DEFINIZIONE DI COSTRUTTO ITERATIVO

- Il costrutto iterativo (o ciclo) permette di eseguire ciclicamente singole istruzioni, o blocchi delle stesse, fino al verificarsi di una o più condizioni.
- Esistono due tipologie di costrutti iterativi:
  - **pre-condizionali**: la/le condizioni di terminazione del ciclo vengono valutate **prima** di eseguire il blocco di istruzioni interno al ciclo stesso;
  - **post-condizionali**: la/le condizioni di terminazione del ciclo vengono valutate **dopo** aver eseguito il blocco di istruzioni interno al ciclo stesso.

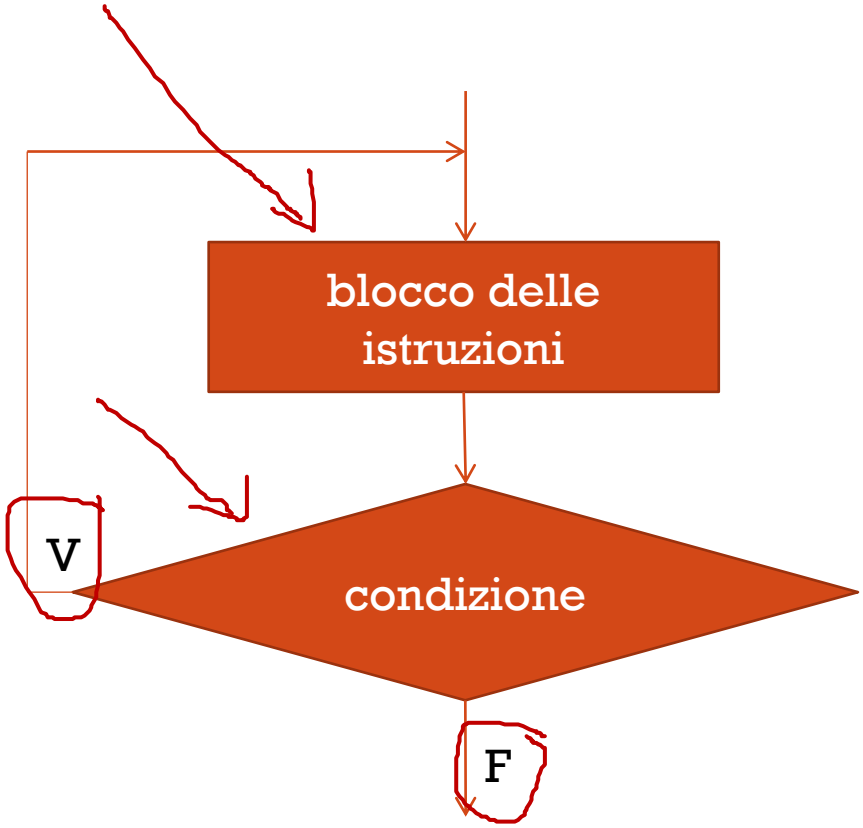
CICLO PRE-CONDIZIONALE



- Se la condizione è "*vera*":
  - esegue il blocco delle istruzioni;
  - ritorna a valutare la condizione.
- Se la condizione è "*falsa*":
  - salta il blocco delle istruzioni;
  - esce dal ciclo.

- N.B. Il blocco delle istruzioni potrebbe non venire **mai** eseguito.

CICLO POST-CONDIZIONALE



- Viene eseguito il blocco delle istruzioni.
- Se la condizione è "*vera*":
  - eseguire il blocco delle istruzioni;
  - valuta la condizione.
- Se la condizione è "*falsa*":
  - esce dal ciclo.

- N.B. Il blocco delle istruzioni viene sempre eseguito **almeno una volta**.

ISTRUZIONE "WHILE"

- L'istruzione "*while*" permette di creare un ciclo di tipo pre-condizionale.

```
while(condizione)
{
  istruzione_1;
  …
  istruzione_n;
}
```

- se "*condizione*" è vera si eseguono "*istruzione_1, …, "istruzione_n*" e poi si torna a valutare la condizione;
- se "*condizione*" è falsa si salta il blocco delle istruzioni, si esce dal ciclo e si prosegue con il resto delle istruzioni del programma.

ESEMPIO "WHILE"

- Visualizza, in ordine crescente, tutti i numeri interi da 0 a 99.

```
int i = 0;

while(i <= 99)
{
   printf("%d ", i);
   i++;
}
```


ISTRUZIONE "FOR"

- L'istruzione "*for*" permette di creare un ciclo di tipo pre-condizionale (compatto).

```
for(inizializzazione; condizione; incremento)
{
  istruzione_1;
  …
  istruzione_n;
}
```

- "*condizione*":
  - se "*condizione*" è vera si eseguono "*istruzione_1, …, "istruzione_n*" e poi si torna a valutare la condizione;
  - se "*condizione*" è falsa si salta il blocco delle istruzioni, si esce dal ciclo e si prosegue con il resto delle istruzioni del programma.

ISTRUZIONE "FOR"

- L'istruzione "*for*" permette di creare un ciclo di tipo pre-condizionale (compatto).

```
for(inizializzazione; condizione; incremento)
{
  istruzione_1;
  …
  istruzione_n;
}
```

- "*inizializzazione*":
  - è un'espressione che viene eseguita, come prima istruzione del ciclo, una sola volta, alla prima esecuzione dell'istruzione "*for*";
  - viene normalmente utilizzata per inizializzare la variabile che controlla il ciclo.

ISTRUZIONE "FOR"

- L'istruzione "*for*" permette di creare un ciclo di tipo pre-condizionale (compatto).

```
for(inizializzazione; condizione; incremento)
{
  istruzione_1;
  ...
  istruzione_n;
}
```

- "*incremento*":
  - è un'espressione che viene eseguita, come ultima istruzione del ciclo, prima che venga rivalutata la condizione;
  - viene normalmente utilizzato per incrementare/decrementare la variabile che controlla il ciclo.

FOR VS WHILE

FOR

```
for(inizializzazione; condizione; incremento)
{
  istruzione_1;
  …
  istruzione_n;
}
```

WHILE

```
inizializzazione;

while(condizione)
{
  istruzione_1;
  …
  istruzione_n;

  incremento;
}
```

ESEMPIO "FOR"

- Visualizza, in ordine crescente, tutti i numeri interi da 0 a 99.

```
int i;

for(i = 0; i <= 99; i++)
{
  printf("%d ", i);
}
```

ISTRUZIONE "DO - WHILE"

- L'istruzione "*do - while*" permette di creare un ciclo di tipo post-condizionale.

```
do
{
  istruzione_1;
  …
  istruzione_n;
}
while(condizione);
```

- se "*condizione*" è vera si ritorna all'inizio del ciclo, si ri-eseguono "*istruzione_1, …, "istruzione_n*" e poi si torna a valutare la condizione;
- se "*condizione*" è falsa si esce dal ciclo e si prosegue con il resto delle istruzioni del programma.

ESEMPIO "DO - WHILE"

- Visualizza, in ordine crescente, tutti i numeri interi da 0 a 99.

```
int i = 0;

do
{
   printf("%d ", i);
   i++;
}
while(i <= 99);
```

ESERCIZI

- **Esercizio 1:** scrivere un programma che legga 10 numeri interi e ne visualizzi il valore assoluto.
- **Esercizio 2:** scrivere un programma che legga N numeri interi positivi minori di 1000 e ne visualizzi il massimo, il minimo e la media.
- **Esercizio 3:** scrivere un programma che visualizzi i primi 100 numeri interi positivi pari.
- **Esercizio 4:** scrivere un programma che data una sequenza di numeri interi positivi terminata dallo zero. Visualizzi quanti tra i numeri inseriti sono minori, maggiori o uguali ad un numero X inserito da tastiera.

ESERCIZI

- **Esercizio 5:** scrivere un programma che verifichi se il valore inserito (intero positivo) dato è un numero primo.
- **Esercizio 6:** scrivere un programma che calcoli e visualizzi la somma dei primi N numeri primi.
- **Esercizio 7:** scrivere un programma che legga una sequenza di numeri interi, la sequenza termina quando viene inserito un numero minore del precedente. Il programma deve visualizzare quanti numeri sono stati inseriti.