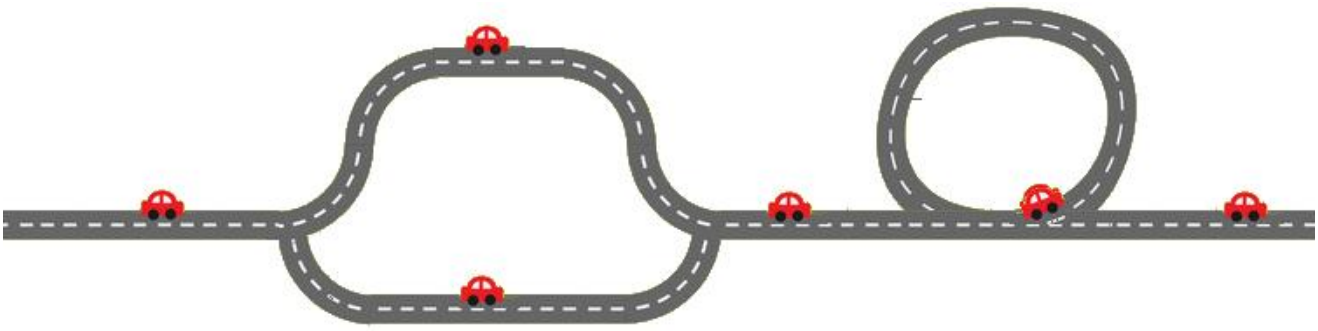
# ISTRUZIONI DI CONTROLLO

Le strutture di controllo sono delle particolari istruzioni, tipiche dei linguaggi imperativi, che permettono di eseguire delle istruzioni secondo determinate condizioni.

**Teorema di Bohm-Jacopini** Gli informatici Corrado Bohm e Giuseppe Jacopini, nel 1966 enunciarono il seguente teorema: Ogni algoritmo può essere implementato utilizzando tre sole strutture:

- **Sequenza**: blocchi di codice procedurale le cui istruzioni vengono eseguite nella stessa sequenza con le quali sono state scritte nel codice sorgente. Queste possono contenere strutture di controllo condizionali e/o iterative.
- **Selezione o scelta binaria**: permettono di specificare due rami o blocchi (uno del *vero* ed uno del *falso*) di codice, di cui solo uno verrà eseguito in base al risultato booleano dell'espressione condizionale (vero/falso).
- **Iterazione o ciclo**: eseguono lo stesso blocco di codice, in modo ripetitivo, fino a quando l'espressione condizionale è vera.

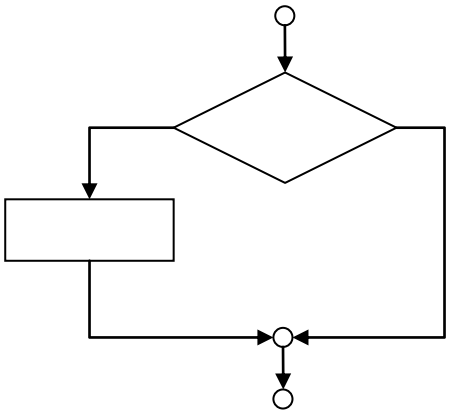
Similitudini con le vie di comunicazione:



*Un'automobile può procedere per una strada sulla quale può incontrare un bivio (SCELTA): a questo punto può scegliere in quale direzione proseguire, naturalmente le due strade ad un certo punto si incontrano nuovamente. Continuando può decidere di entrare in un circuito (CICLO) o proseguire senza percorrerlo.*

Le parentesi graffe { … } si utilizzano quando si deve eseguire più di una istruzione

# SELEZIONE O SCELTA BINARIA



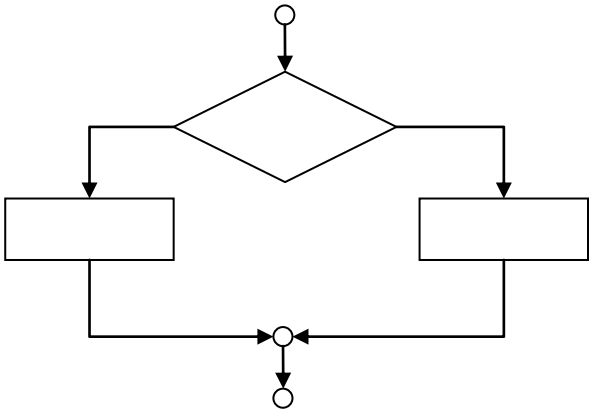
## IF

If <espressione>
    istruzione

If <espressione>
{
    istruz. 1
    Istruz. 2
    ...
}

Esempio:

```
if (a<20)
{
   cout << "minore di 20" << endl;
}
```

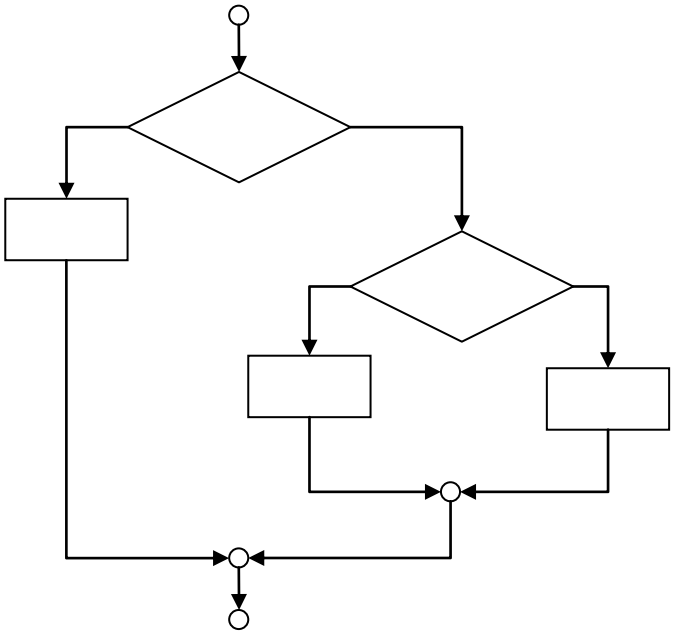


## IF - ELSE

If <espressione>
{
    istruz. 1
    Istruz. 2
    ...
}
Else
    {
      istruz. 1
      Istruz. 2
      ...
}

Esempio:

```
If (n1==n2)
{
   cout << "uguali" << endl;
}
else
{
   cout << "diversi" << endl;
}
```

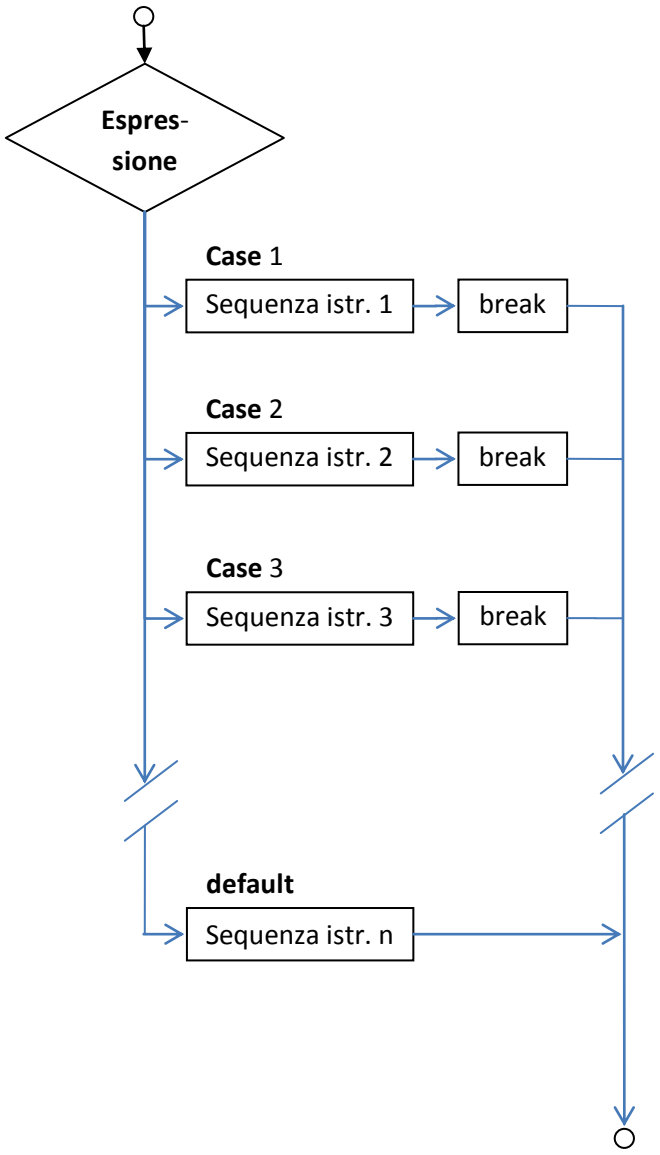


## IF – ELSEIF - ELSE

```
If <espressione>
{
      istruz. 1
      Istruz. 2
       ...
}
Elseif <espressione>
        {
          istruz. 1
          Istruz. 2
           ...
        }
      Else
        {
          istruz. 1
          Istruz. 2
           ...
        }
```

Esempio:

```
If (n <0)
{
   cout << "negativo" << endl;
}
else if (n==0)
    {
      cout << "zero" << endl;
    }
    else
    {
      cout << "positivo" << endl;
    }
```




## SWITCH CASE

```
switch (<espressione intera>)
{
    case (<valore costante 1>):
     <sequenza di istruzioni 1>
     break;

    case (<valore costante 2>)
     <sequenza di istruzioni 2>
     break;

    ...

    case (<valore costante N>)
     <sequenza di istruzioni N>
     break;

    default:
     <sequenza di istruzioni N+1>
}
```

Esempio:

```
switch(n)
{
  case 1:
     cout<<"1"<<endl; break;

  case 2:
     cout<<"2"<<endl; break;

  case 3:
     cout<<"3"<<endl; break;

  default: //opzionale
     cout<<"Altro numero."<<endl;
}
```

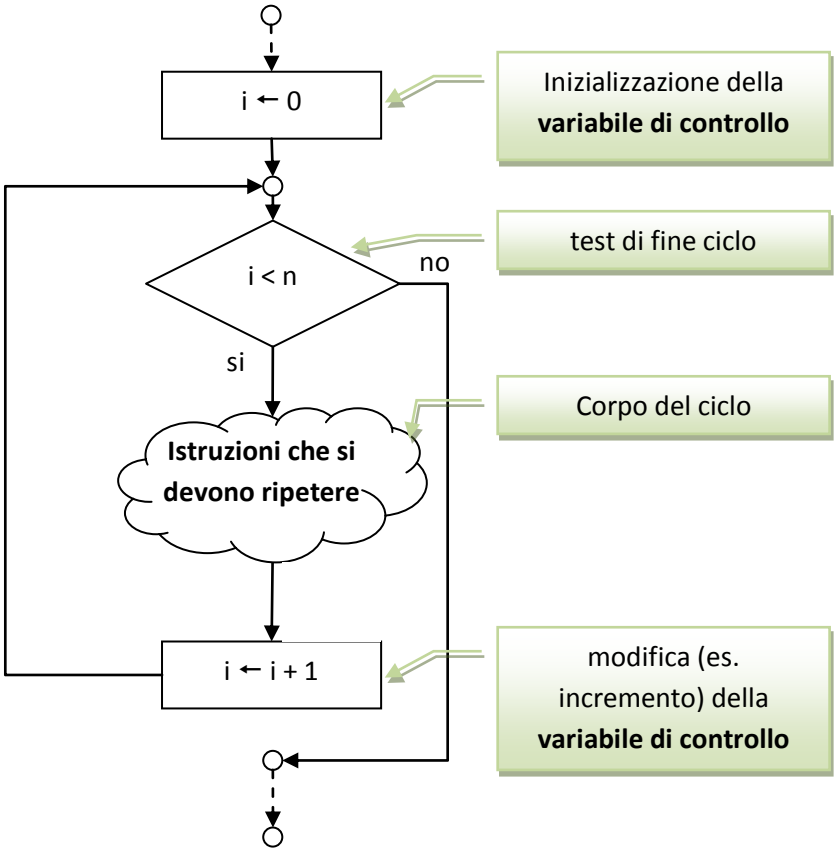
Lo switch ha una struttura a cascata, cioè se eseguo il primo caso (avendo sotto degli altri) e non metto un `break` per uscire, continua ad eseguire anche le istruzioni successive che fanno parte di altre casistiche.

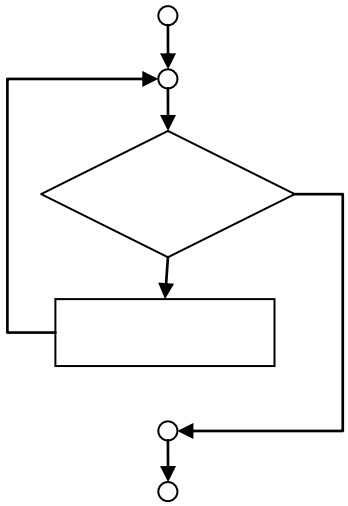
# ITERATIZIONE o CICLO

La struttura di controllo iterativa determina l'esecuzione di una porzione di programma ripetuta per un certo numero di volte.

Il ciclo si dice *definito* quando a priori si conosce il numero di volte che si ripeterá, *indefinito* nel caso contrario.

Il test determina fino a quando le istruzioni del corpo del ciclo devono essere ripetute





## FOR … TO … NEXT

**For (** <var>=val.iniziale, condizione, increm.**)**
{
    Istruz. 1
    Istruz. 2
    …
}

**SI RIPETE PER FALSO**

Esempio:

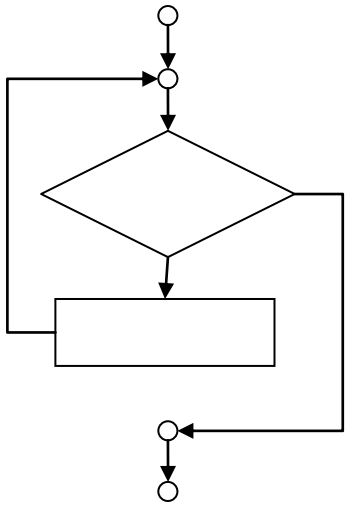
```
int totale = 0; // Questa variabile
conterrà la somma totale

// Il ciclo seguente aggiunge i numeri
da 1 a 10 alla variabile totale
for (int i=1; i <= 10; i++)
{
    totale += i;
}
```

- Come valore_iniziale abbiamo l'espressione int i=1, con cui oltre a indicare la variabile i come variabile contatore inizializzata a 1, 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 (l'istruzione i++ serve per incrementare di un'unità un intero).

## CICLI INDEFINITI

Un ciclo è **indefinito** quando non è possibile conoscere a priori quante volte verrà eseguito



### WHILE

**While (condizione);**
{
    Istruz. 1
    Istruz. 2
    …
}

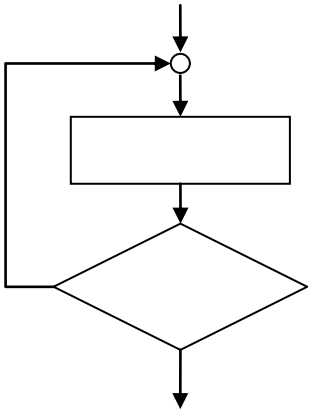
Nel caso in cui la condizione sia subito falsa allora il ciclo non verrà eseguito neanche una volta.

**SI RIPETE PER VERO**

Esempio:

```
//calcolo della media aritmetica
int i=0,somma=0, media, n, a;

cin>>n;
while(i<n)
{
  cin>>a;
  somma=+a;
  i++;
}
media=somma/n;
```



## DO... WHILE

```
Do
{
      Istruz. 1
      Istruz. 2
      …
} While (condizione);
```

**SI RIPETE PER VERO**

Le istruzioni del corpo del ciclo sono eseguite **almeno una volta**, indipendentemente dal verificarsi o meno delle condizioni.

Esempio:

```
//leggere un numero maggiore di 0
do
{
  Cin>>n;
  if (c<=0)
  {
    Cout<<"numero non positivo");
  }
} while (n<=0));
```

## Break e Continue

- break – esce dal ciclo o dallo switch;
- continue – salta una iterazione del ciclo senza interromperlo;