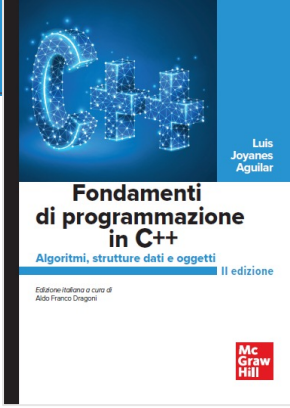
Capitolo 4

La programmazione strutturata

pag. 89-122

Presenta: Prof. Misael Mongiovì



strutture di controllo

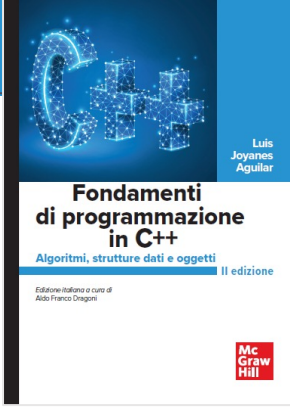
- determinano il flusso d'esecuzione di un programma
- permettono di combinare più istruzioni in una semplice unità logica con un punto d'ingresso ed uno d'uscita
- tre tipi:

1) sequenza

```
{
istruzione 1;
istruzione 2;
…
istruzione n;
}
```

2) selezione

3) ripetizione



sequenza: regole di visibilità

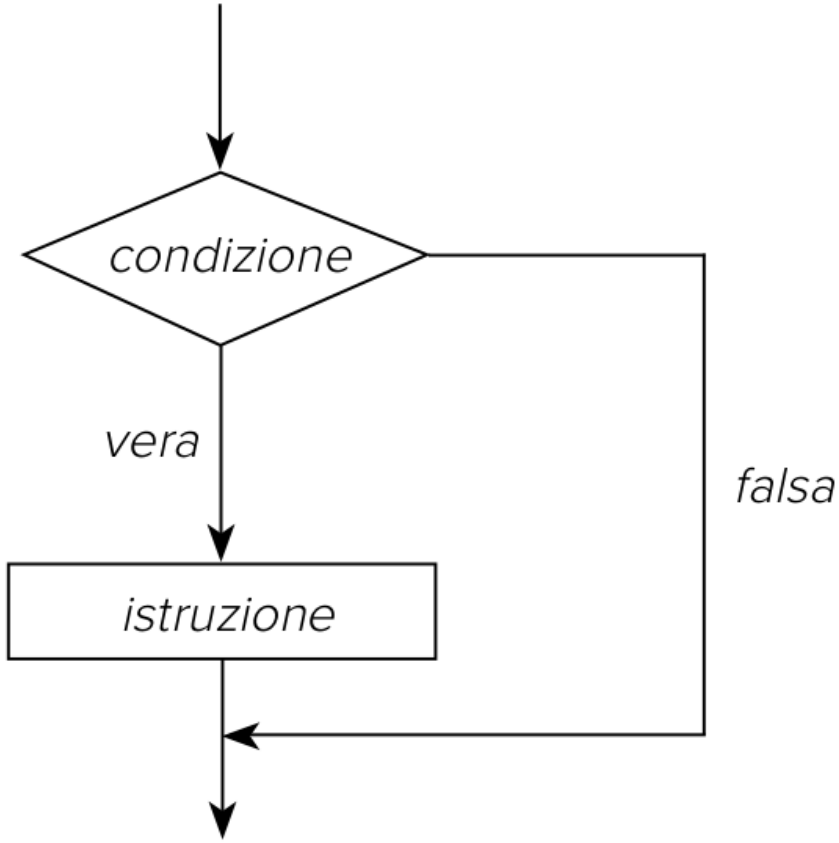
```
int main() {

   int x = y*2;

   int y = 3;

}
```

```
int main() {

   int y = 3;

   int x = y*2;

}
```

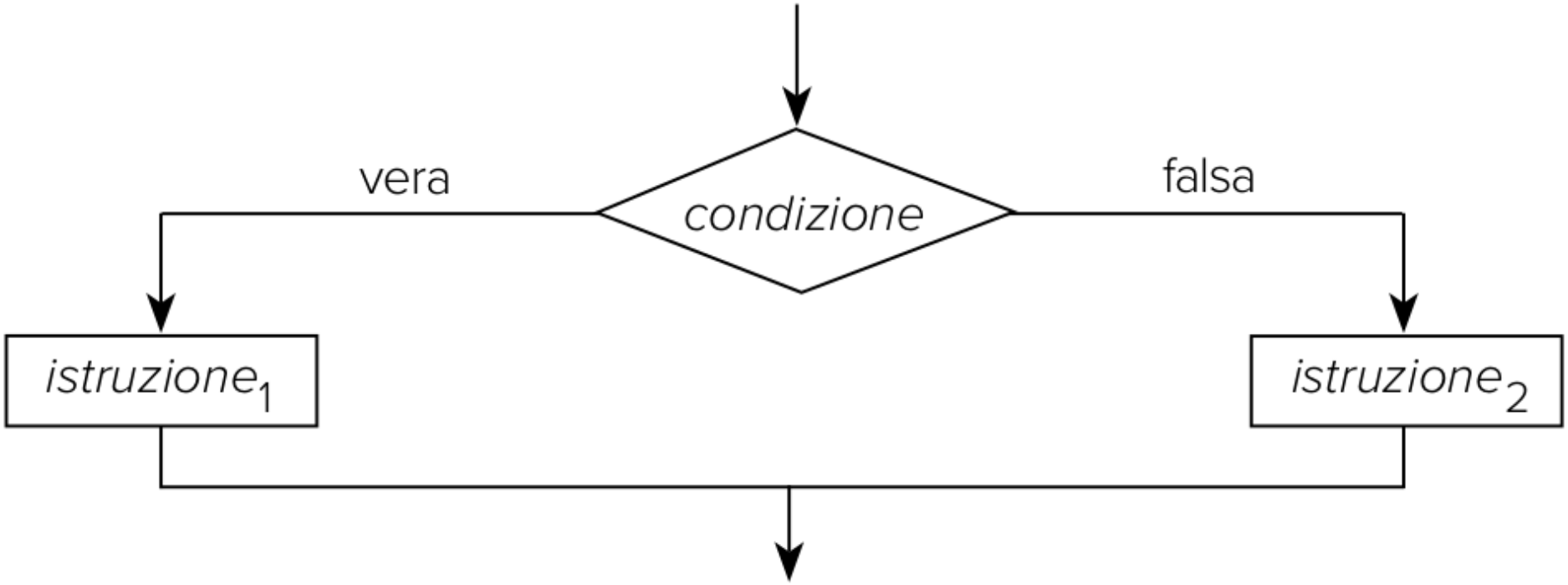
```
int main() {
   int x;
   {
        int y = 3;
   }
   x = y;
}
```

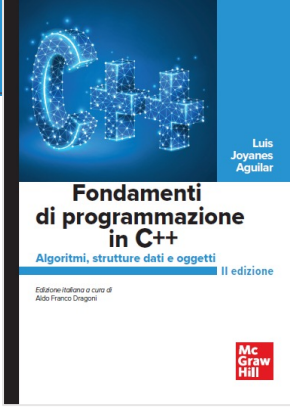
```
int main() {
   int x;
   {
        int y = 3;
x = y;
   }
}
```


```
if (condizione)
    istruzione
```




```
if (cond) istr1 else
          istr2
```





istruzioni `if else` annidate

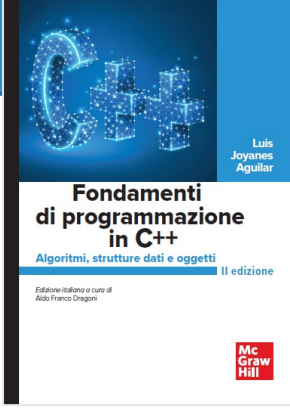
- se abbiamo più di due alternative possiamo "annidare" più istruzioni `if` una dentro l'altra

```
if (voto < 6) valutazione = "sufficiente";
 else
 if (voto < 7) valutazione = "buono";
  else
  if (voto < 8) valutazione = "distinto";
   else valutazione = "ottimo";
```

- l'`else` si riferisce sempre all'`if` più vicino

```
if (eta > 18) if (reato) punibile=true; else punibile=false;
```

```
if (eta > 18) {if (reato) punibile=true;} else minorenne=true;
```

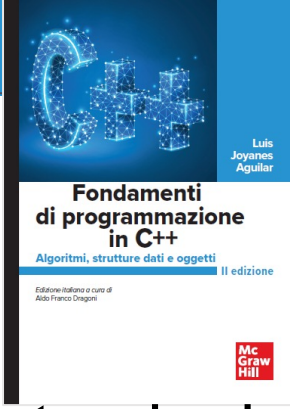


istruzione `switch`

sostituisce gli if annidati

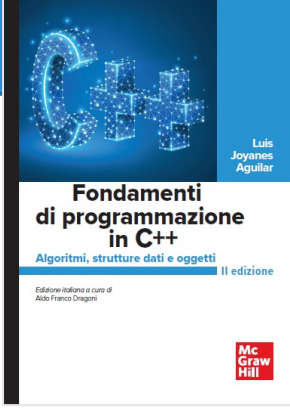
```
switch (selettore)
{
  case etichetta_1: istruzione_1; break;
  case etichetta_2: istruzione_2; break;
    ...
  case etichetta_n: istruzione_n; break;
  default: istruzione_default; // opzionale
}
```

opzionale



le strutture cicliche

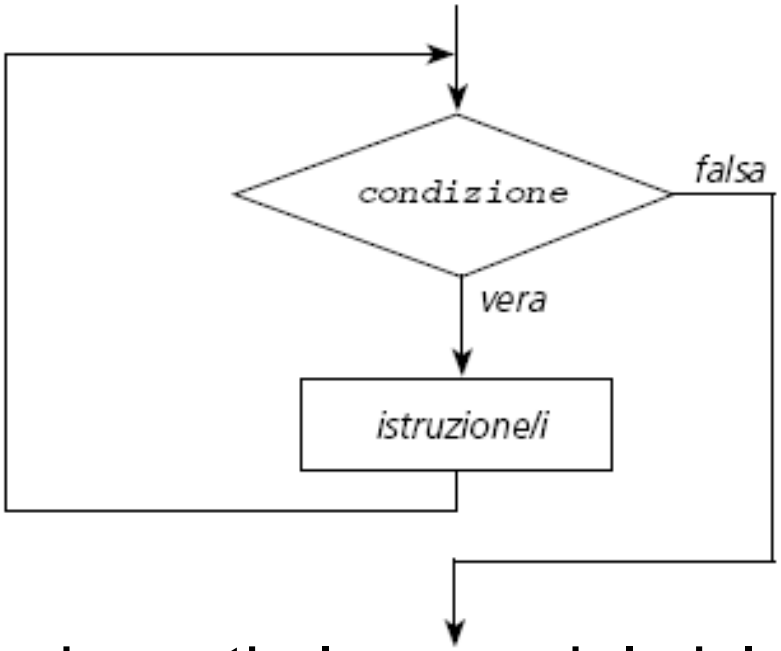
- ***ciclo***: ripetizione controllata di una sequenza di istruzioni (***corpo*** del ciclo)
- ***iterazione***: ogni ripetizione del ciclo
- ***condizione***: espressione booleana che prima o poi dovrà diventare `false` (a meno che non si voglia avviare una ripetizione all'infinito)
- il corpo può essere un'istruzione singola o un'istruzione composta (blocco)



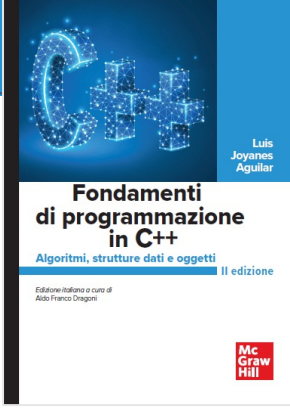
le strutture cicliche

- ripetizione controllata da un *contatore*
  - ripetizione definita (è noto quante volte il ciclo sarà eseguito)
  - variabile di controllo usata per contare le ripetizioni
- ripetizione controllata da una *sentinella*
  - ripetizione indefinita (usata quando il numero di ripetizioni non è noto a priori)
  - il termine del ciclo è indicato dal valore della sentinella

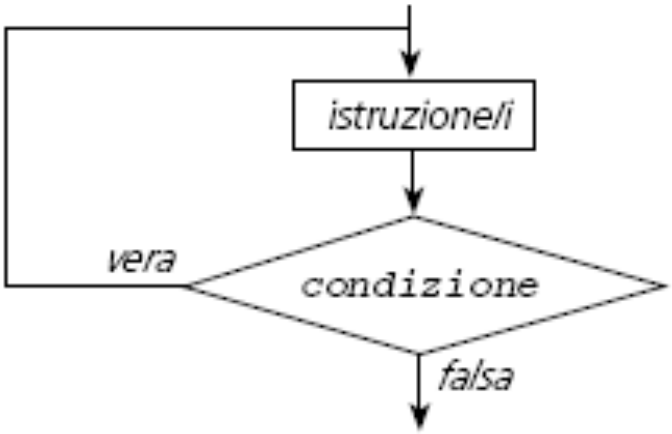
`while (cond) istruzione`



ha la `cond` posta davanti al corpo del ciclo; questo significa che si valuta *prima* la `cond` e poi si esegue eventualmente il corpo del ciclo `istruzione`

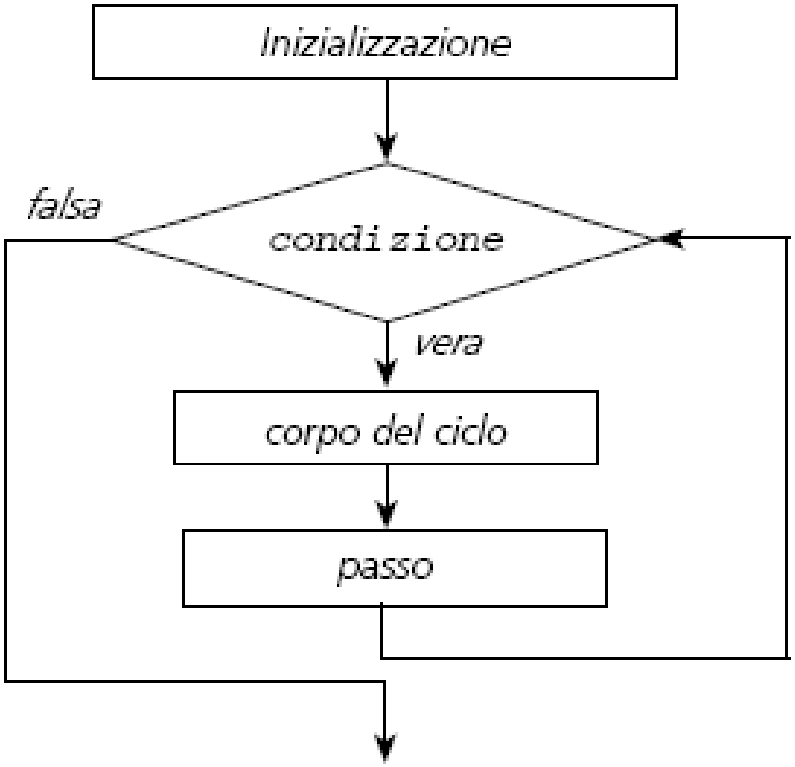


do *istruzione* while (*cond*)



ha la `cond` posta dopo il corpo del ciclo; questo significa che si esegue almeno una volta `istruzione` e poi si valuta `cond` per eventualmente ripeterla


```
for (iniz; cond; passo)istruzione
```



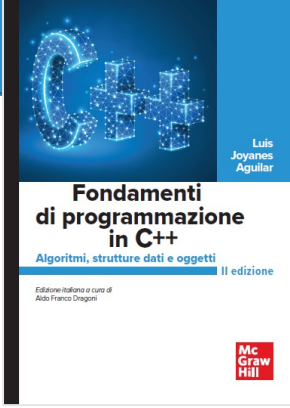
equivale a:

```
inizi;
while (cond)
{
  istruzione;
  passo;
}
```

cicli annidati

- si possono *annidare* cicli uno dentro l'altro

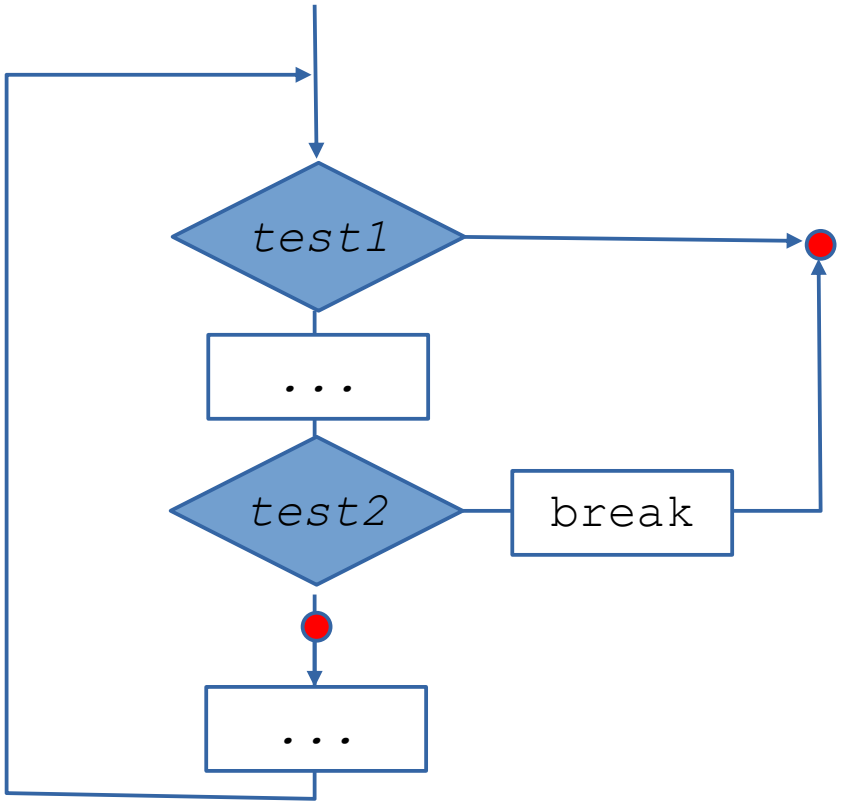
```
do
{
  while (condizione_ciclo_2)
  {
    for (inizializzazione; condizione_ciclo_3; passo)
    {
      ....
    }
  }
} while (condizione_ciclo_1)
```



confronto fra `while`, `do while` e `for`

`while`	È particolarmente usato quando l'iterazione è controllata da sentinella. Poiché il corpo del ciclo può non essere eseguito, si utilizza quando si vuole saltare all'istruzione successiva se la condizione è falsa in partenza.
`for`	È particolarmente usato quando l'iterazione è controllata da contatore. La condizione di controllo precede l'esecuzione del corpo del ciclo.
`do while`	È usato quando si deve eseguire almeno un'iterazione. La condizione di controllo segue l'esecuzione del corpo del ciclo.

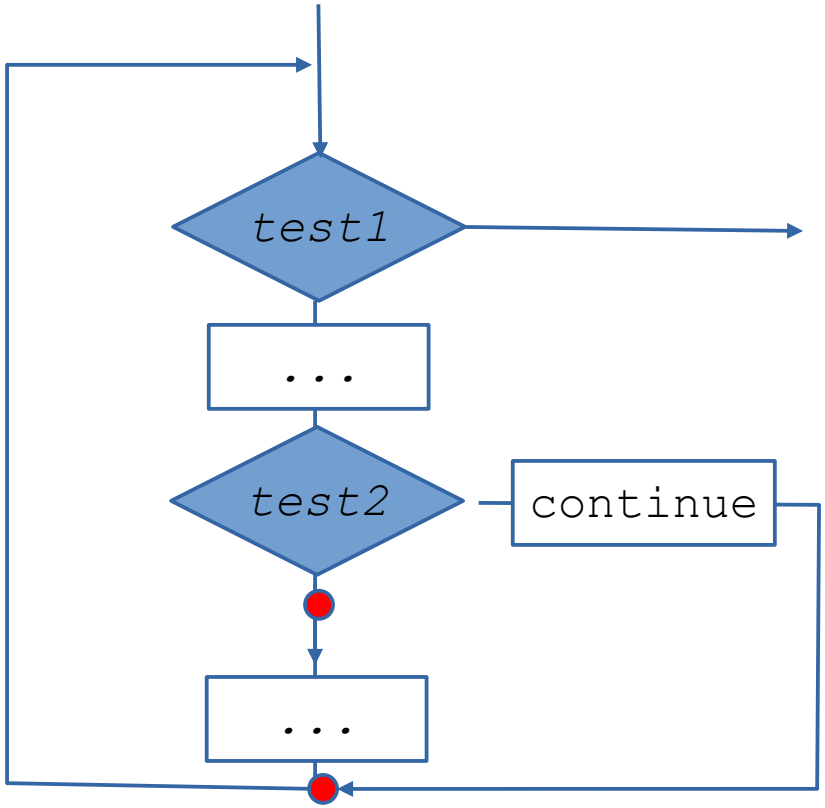
istruzioni di salto: `break`




```
while (test1)
{...
 if(test2) break;
 ...
}
```

- si noti che le 2 uscite dell'`if` che controlla il `break` non si ricongiungono!

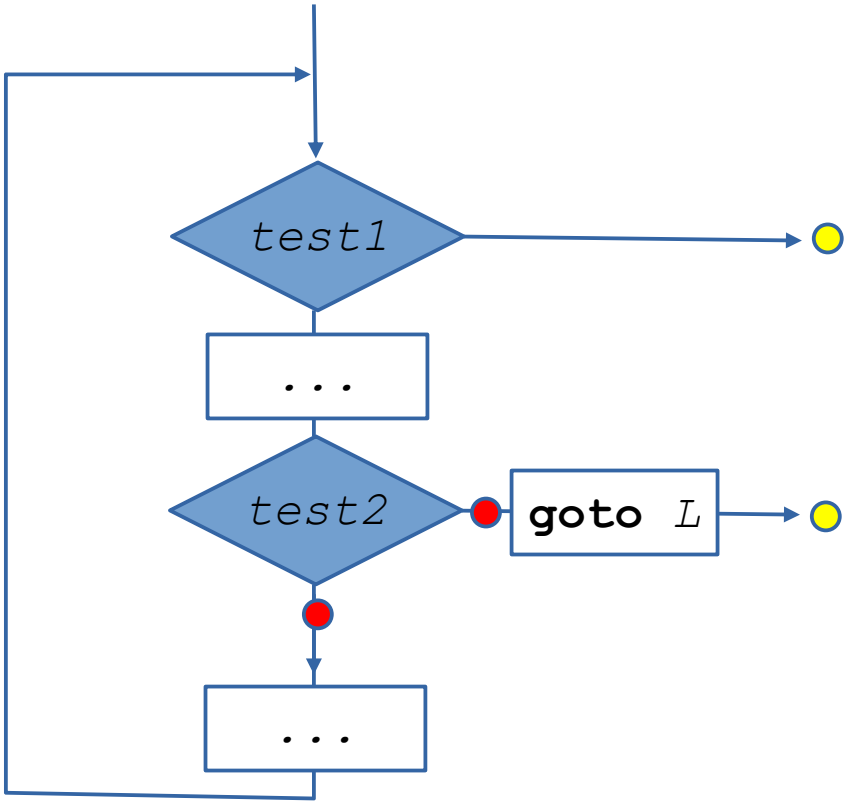
istruzioni di salto: `continue`




```
while (test1)
{...
 if(test2) continue;
 ...
}
```

- si noti che le 2 uscite dell'`if` che controlla il `continue` non si ricongiungono!

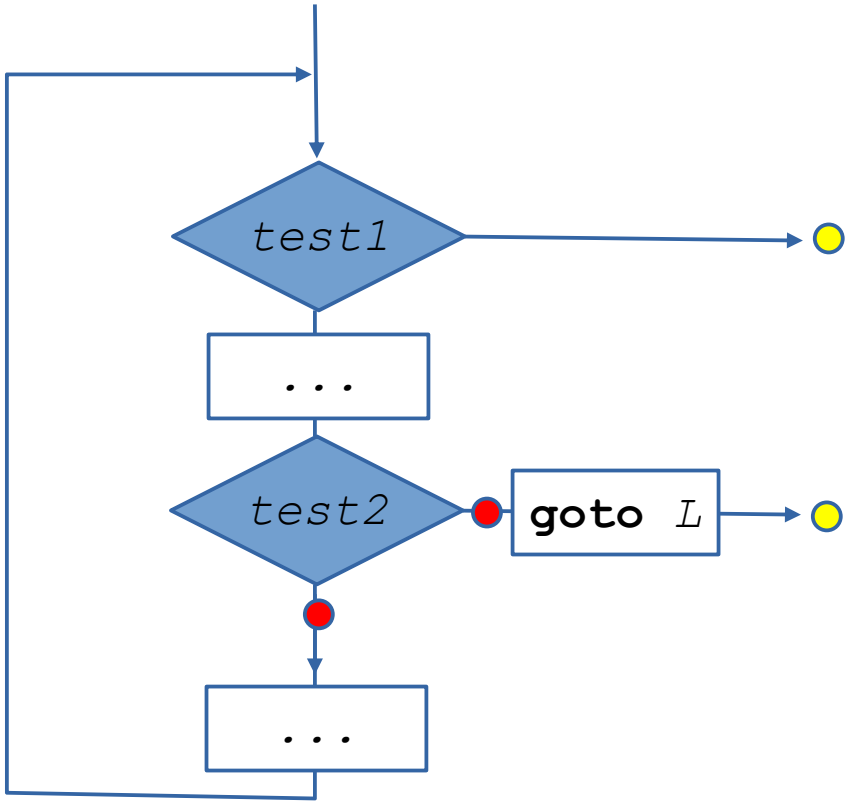
istruzioni di salto: `goto`




```
while (test1)
{...
 if(test2) goto L;
 ...
}
...
L: istruzione
```

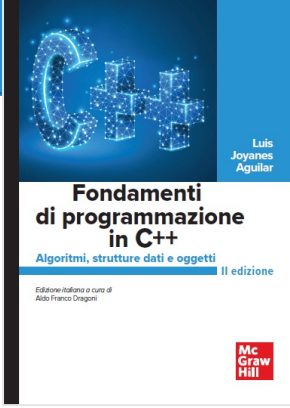
- si noti che le 2 uscite dell'`if` che controlla il `break` non si ricongiungono!
- si noti che il ciclo ha 2 uscite!

istruzioni di salto: `goto`




```
while (test1)
{...
 if(test2) goto L;
 ...
}
...
L: istruzione
```

- si noti che le 2 uscite dell'`if` che controlla il `break` non si ricongiungono!
- si noti che il ciclo ha 2 uscite!



istruzioni di salto: esempi

```
int x=0;
int main()
{
    while(x < 10)
    {
        x++;
//      if (x == 5) break;
//      if (x == 5) continue;
//      if (x == 5) goto paperino;
        cout << "x vale " << x << endl;
    }
    cout << "ho finito di contare \n";
    paperino: cout << "perché ho finito le dita \n";
    return 0;
}
```

```
x vale 1
x vale 2
x vale 3
x vale 4
x vale 5
x vale 6
x vale 7
x vale 8
x vale 9
x vale 10
ho finito di contare
perché ho finito le dita
```

```
int x=0;
int main()
{
    while(x < 10)
    {
        x++;
        if (x == 5) break;
//      if (x == 5) continue;
//      if (x == 5) goto paperino;
        cout << "x vale " << x << endl;
    }
    cout << "ho finito di contare \n";
    paperino: cout << "perché ho finito le dita \n";
    return 0;
}
```

```
x vale 1
x vale 2
x vale 3
x vale 4
ho finito di contare
perché ho finito le dita
```

istruzioni di salto: esempi

```
int x=0;
int main()
{
    while(x < 10)
    {
        x++;
//      if (x == 5) break;
        if (x == 5) continue;
//      if (x == 5) goto paperino;
        cout << "x vale " << x << endl;
    }
    cout << "ho finito di contare \n";
    paperino: cout << "perché ho finito le dita \n";
    return 0;
}
```

```
x vale 1
x vale 2
x vale 3
x vale 4
x vale 6
x vale 7
x vale 8
x vale 9
x vale 10
ho finito di contare
perché ho finito le dita
```

```
int x=0;
int main()
{
    while(x < 10)
    {
        x++;
//      if (x == 5) break;
//      if (x == 5) continue;
        if (x == 5) goto paperino;
        cout << "x vale " << x << endl;
    }
    cout << "ho finito di contare \n";
    paperino: cout << "perché ho finito le dita \n";
    return 0;
}
```

```
x vale 1
x vale 2
x vale 3
x vale 4
perché ho finito le dita
```