Le espressioni logiche

All'interno delle *espressioni relazionali*, che utilizziamo ad esempio all'interno di *while()* o di *if(),* è possibile inserire più di una condizione, inserendo più espressioni relazionali.

Questo è possibile grazie alle **espressioni logiche**.

Esistono tre tipi di operazioni logiche possibili nel C++, **AND**, **OR** e **NOT**.

Vediamo i rispettivi operatori logici:

```
AND: &&
OR: ||
NOT: !
```

Vediamo il loro utilizzo uno per volta.

L'operatore **AND**, che in italiano può essere tradotto con la congiunzione *"e"*, serve per verificare che **tutte** le espressioni siano vere.
Ad esempio:

```
int varInt = 20;

if(varInt > 5 && varInt &lt 15)
{
//...codice...
}
```

Questo *if* controlla se varInt è maggiore di 5 e minore di 15.
La prima condizione è vera mentre la seconda è falsa, quindi il codice all'interno del blocco non viene eseguito.
Se la variabile varInt fosse stata ad esempio uguale a 10, entrambe le condizioni sarebbero state confermate, ed il codice sarebbe stato eseguito:

```
se(espressione1 e espressione2) si verificano...
```

L'operatore **OR**, che può essere tradotto come *"o"*, serve invece per verificare che **almeno una** delle istruzioni sia vera.
Ad esempio:

```
int varInt = 20;

if(varInt > 5 || varInt &lt 15)
{
//...codice...
}
```

Anche in questo caso si verifica una sola delle condizioni, ma avendo utilizzato l'operatore OR, il codice all'interno del blocco viene eseguito.
Perché il codice non sia eseguito è necessario che nessuna delle condizioni si verifichi.

L'operatore **NOT**, infine, fa sì che il blocco di codice sia eseguito se la condizione specificata dall'espressione relazionale **non** si verifica.
Ad esempio:

```
int varInt = 0;

if( !(varInt == 0) )
{
//...codice...
}
```

L'espressione relazionale dice di controllare *se varInt è uguale a 0*.
Tuttavia il punto esclamativo che precede l'espressione dice all'if di eseguire il blocco di codice solo se la condizione **non** si verifica, e cioè se varInt **non è** uguale a 0.
In gergo tecnico si dice che l'operatore NOT **nega** l'espressione.

L'operatore NOT ci sarà molto utile per controllare il valore restituito dalle funzioni.

Ovviamente non si è limitati all'utilizzo di due sole espressioni relazionali:

```
if(varInt > 5 || varInt &lt 15 || varInt != 7 || varInt != 9 )
```

Ne tantomeno si è limitati ad utilizzare un singolo operatore logico per espressione:

```
if(varInt > 5 || varInt &lt 15 && varInt != 8)
```

Sono entrambe operazioni legittime.