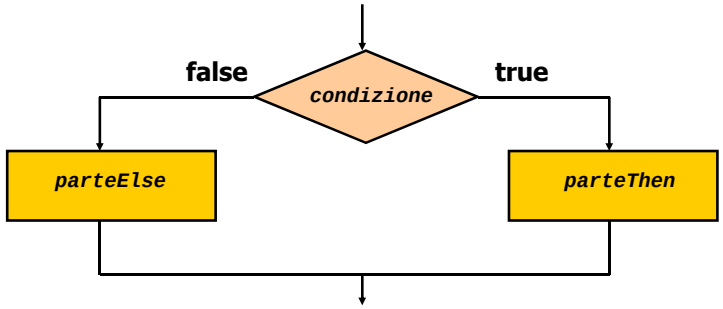
Istruzione condizionale if-then-else

- L'**istruzione condizionale if-then-else** permette di scegliere le azioni alternative da intraprendere in base al valore di una espressione booleana
- Schema generale dell'istruzione if-then-else:
  `if (condizione) parteThen else parteElse`
  - ***`condizione`*** e' una espressione booleana. ***`parteThen`*** e ***`parteElse`*** sono dei blocchi di istruzioni (possibilmente una singola istruzione)
  - L'istruzione if-then-else viene eseguita nel seguente modo: viene valutata la ***`condizione`***; se il risultato e' **`true`** allora viene eseguita la ***`parteThen`*** e la ***`parteElse`*** viene saltata; se il risultato e' invece **`false`**, viene saltata la ***`parteThen`*** e viene eseguita la ***`parteElse`***. Naturalmente, dopo l'if-then-else, indipendentemente dal ramo seguito, l'esecuzione continua con l'istruzione successiva all'if-then-else

Istruzione condizionale if-then-else



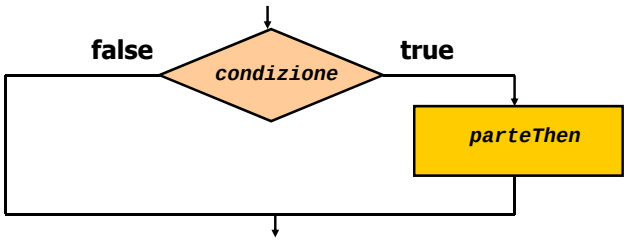
Semplice esempio:

```
if (numero >= 0)
      cout << "Maggiore o uguale a zero";
else cout << "Negativo";
```

Istruzione condizionale if-then

- Il ramo ***`else`*** e' **opzionale**, cioe` si puo' omettere. Quindi, l'istruzione condizionale puo' essere usata in una versione "if-then": se ***`condizione`*** si valuta a **`true`** allora viene eseguita la ***`parteThen`***, se invece si valuta a **`false`**, l'istruzione if-then non ha alcun effetto, e viene quindi eseguita l'istruzione successiva
- **Esempio:**

```
if ((giorno == 25) && (mese == 12))
    cout << "E' Natale!";
```



Esempio

Problema: scrivere un programma che date tre variabili intere x, y e z stampa il valore minore tra quelli memorizzati nelle tre variabili.

```
#include<iostream>
using namespace std;

main(){
  int x = 5;  int y = -2; int z = 3;
  if (x < y) {
    if (x < z) { cout << x; }
    else { cout << z; }
  }
  else {
    if (y < z) { cout << y; }
    else { cout << z; }
  }
}
```

Esercizi

- Correggere il seguente frammento di codice in modo che fornisca il risultato inteso

```
if (totale == max) {
   if (totale < somma) {cout << "totale e' uguale a max e minore di somma"; }

   else {cout << "totale non e' uguale a max";}
}
```

- Il seguente frammento di codice potrebbe essere compilato senza errori? Perche'?

```
if (lunghezza = min) { cout << "Non si puo' diminuire la lunghezza."; }
```

Esercizi (soluzioni)

- Correggere il seguente frammento di codice in modo che fornisca il risultato inteso

```
if (totale == max) {
   if (totale < somma) {cout << "totale e' uguale a max e minore di somma";}

   else {cout << "totale non e' minore di somma";}

}
```

- Il seguente frammento di codice potrebbe essere compilato senza errori? Perche'?

```
if (lunghezza = min) { cout << "Non si puo' diminuire la lunghezza."; }
```

C'e' un errore. La versione corretta e'`(lunghezza == min)`

Istruzioni di ripetizione

- Istruzioni di ripetizione: prevedono di ripetere uno stesso blocco di operazioni piu' volte
- Tali ripetizioni vengono formulate come cicli (o loop). Ci sono due categorie di cicli:
  - cicli for
  - cicli while

Ciclo `for -- 1`

- Schema generale di un ciclo for

```
for (inizializzazione; controllo; aggiornamento)
{corpo}
```

- La parte ***`inizializzazione`*** viene eseguita una sola volta all'inizio del ciclo. La sequenza di esecuzione delle altre tre parti del ciclo e': *(1)* ***`controllo`***, ***`(2) corpo`***, ***`(3) aggiornamento`***
- La parte ***`inizializzazione`*** introduce e/o inizializza una **variabile di ciclo** (in generale ci potrebbero essere piu` variabili di ciclo) che deve essere testata nella parte ***`controllo`*** e aggiornata nella parte ***`aggiornamento`***.

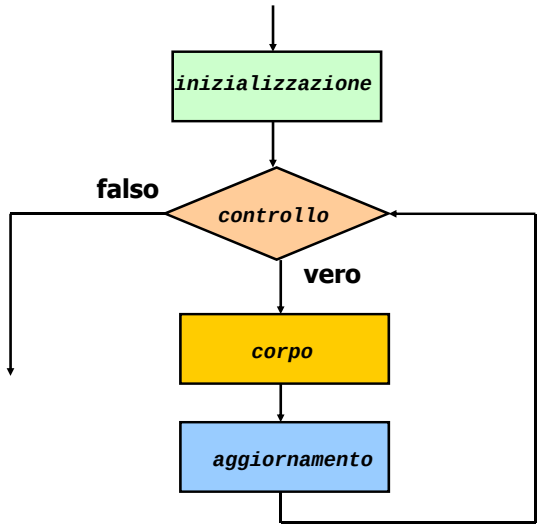
Ciclo `for` -- 2

- Schema generale di un ciclo for

  for (inizializzazione; controllo; aggiornamento) {corpo}

- La parte ***controllo*** consiste in una espressione booleana che fornisce la **condizione di terminazione** del ciclo, basato sul valore corrente (ovvero a quel punto della ripetizione del ciclo) della variabile di ciclo. Questo confronto viene eseguito ad ogni iterazione del ciclo
- La parte ***aggiornamento*** e' generalmente costituita da **assegnazioni che modificano il valore della variabile di ciclo**, ogni volta che il ciclo viene eseguito
- Il ***corpo*** del ciclo e' costituito da una sequenza di istruzioni, che viene eseguita dopo il controllo

Ciclo `for`

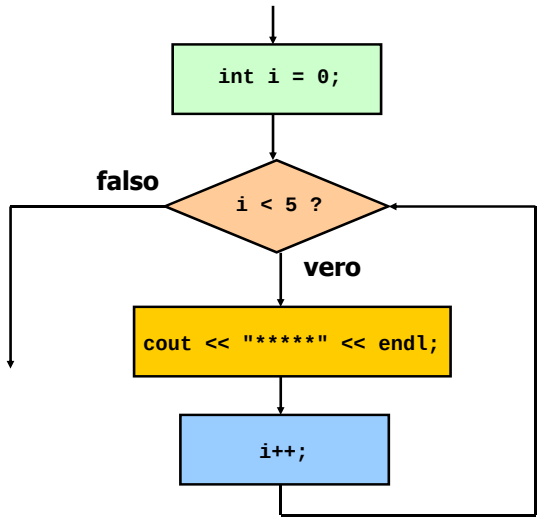


Ciclo `for`: esempio

```
for(int i=0; i < 5; i++) {
  cout << "*****" << endl;
}
```

- **Quante righe di asterischi vengono stampate? Cinque o sei?**
- **`i`** → variabile (intera) del ciclo
- Inizializzazione → **`int i=0`**
- Controllo → **`i<5`**
- Aggiornamento → **`i++`**
- Corpo → **`cout << "*****" << endl;`**
- **Quindi, iteravamente:**
  1) si valuta il test booleano **`i < 5;`** se il test da' **`false`** si esce dal ciclo, altrimenti
  2) si esegue l'unica istruzione del corpo:
     `cout << "*****" << endl;`
  3) si esegue l'aggiornamento della variabile **`i`** incrementandola di 1, ovvero **`i++`**

Ciclo `for`: esempio



Ciclo `for`: esempio

- **Quindi:** con valori di `i` che vanno da 0 a 4 vengono stampate **cinque righe di asterischi**
- Successivamente **`i`** viene incrementato a 5, ed il test 5 < 5 produce **`false`**, e quindi provoca l'uscita dal ciclo

- In generale, il seguente semplice schema di ciclo for

```
for(int i=0; i < n; i++) {
   corpo
}
```

permette di eseguire le istruzioni del corpo **`n`** volte, (**Nota Bene**) se **`corpo`** non influenza il valore della variabile di ciclo **`i`**

Esempio

```
#include<iostream>
using namespace std;

main(){
  for(int i=0; i < 5; i++) {
    cout << i+1 << " *****" << endl;
  }
}
```

Stampa in output:

```
1 *****
2 *****
3 *****
4 *****
5 *****
```

Esercizi

- Quanti asterischi stampa il seguente ciclo?

```
for (int star=9; star<0; star++){cout << '*';}
```

- Cosa stampano in output le seguenti istruzioni?

```
int s = 1;
for (int x=0; x<=5; x++){
       s = s+x;
       cout << s << " ";
     }
```