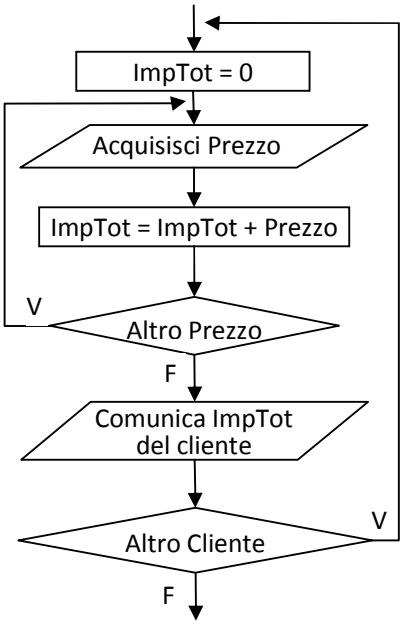
Totalizzatori e contatori

Nelle elaborazioni cicliche sono spesso utilizzati dei **totalizzatori** e dei **contatori**.

- **Totalizzatore**
  Per chiarire meglio il concetto di **totalizzatore** (detto anche **accumulatore**), si pensi alle azioni eseguite dal cassiere di un supermercato quando si presenta un cliente con il proprio carrello pieno di merce. Il cassiere effettua una elaborazione *ciclica* sulla merce acquistata: ogni oggetto viene esaminato per *acquisire il suo prezzo*. Lo scopo della elaborazione è quello di cumulare i prezzi dei prodotti acquistati allo scopo di stabilire il *totale* che il cliente dovrà corrispondere.

  Dal punto di vista informatico si tratta di utilizzare una variabile (nell'esempio potrebbe essere rappresentata dal totalizzatore di cassa) che viene aggiornata per ogni prezzo acquisito: *ogni nuovo prezzo acquisito non deve sostituire il precedente ma aggiungersi ai prezzi già acquisiti precedentemente*. Quando i valori da esaminare terminano, il totalizzatore conterrà il valore cercato. Nell'esempio sopra significa che: finito l'esame dei prodotti acquistati, si potrà presentare al cliente il totale da corrispondere.

  Tale variabile (**totalizzatore**):

  1. **dovrà essere azzerata** *ogni volta che si passa ad un nuovo cliente* (ogni cliente dovrà corrispondere solamente il prezzo dei prodotti che lui acquista e non anche quelli dei clienti che lo precedono)
  2. **si aggiornerà per ogni prodotto esaminato** (ogni nuovo prezzo acquisito verrà cumulato ai precedenti)
  3. finito l'esame dei prodotti acquistati da un cliente, la variabile, **conterrà il valore totale** che quel cliente pagare.

  In generale si può dire che l'uso di un **totalizzatore** prevede i seguenti passi:

  - Inizializzazione totalizzatore
  - Inizio ciclo
  - ...
  - Aggiornamento totalizzatore
  - Fine ciclo
  - Uso del totalizzatore

  La variabile di cui si parla nell'esempio è quella che, nel linguaggio della programmazione, viene definita un **totalizzatore** o **accumulatore**: cioè una variabile nella quale ogni nuovo valore non sostituisce ma si accumula a quelli già presenti in precedenza.

- **Contatore**

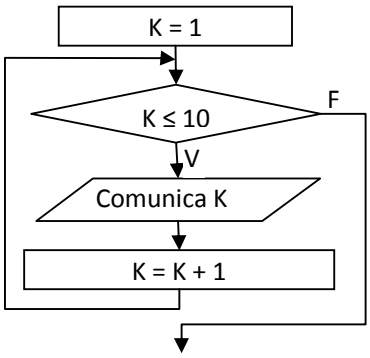
  Se la variabile *si aggiorna sempre di una **quantità costante*** (per esempio viene sempre aggiunta l'unità) viene chiamata **contatore**.

  Una applicazione diffusa dei contatori è quella di controllo delle **elaborazioni iterative a contatore** (o **cicli enumerativi**). Ci sono delle elaborazioni cicliche in cui *è noto il numero degli elementi da elaborare* e, in questi casi, un contatore, che si aggiorna ad ogni elaborazione effettuata (si pensi ad esempio ad un contachilometri di una automobile che si aggiorna in automatico ad ogni chilometro percorso), conteggia gli elementi che vengono trattati mano a mano. Appena il contatore raggiunge la quantità prestabilita, l'elaborazione ha termine.

  In questi casi lo schema generale dell'elaborazione ciclica può assumere questo aspetto:

  - Ricevi Quantità elementi da elaborare
  - **Per** contatore che va da 1 a quantità elementi da elaborare
  - Ricevi elemento
  - elabora elemento
  - **Fine-Per**

  Supponiamo di dover comunicare i primi 10 numeri interi partendo da 1. Il nostro contatore dovrà assumere i valori da 1 a 10. Viene inizializzato con il valore 1 e ogni volta sarà incrementato di una unità fino ad arrivare a 10. Raggiunto 10 si esce dal ciclo.



  Per risolvere questo problema si può utilizzare il ciclo **Per** (in C++ : **for** …)
  I passi sono i seguenti:

  - **Ciclo Per** Contatore che va dal *valore di partenza* al *valore finale*
  - ...
  - Incremento contatore
  - **Fine-Per**

```
....
for ( K = 1; K <= 10; K++)
   {cout << K << endl;
   }
……
```

  Pur essendo il ciclo **Per** quello più utilizzato per i cicli a contatore, tuttavia si possono utilizzare anche i cicli **Mentre** (in C++ : **while** …) o **Esegui … finché** (in C++ : **do** … **while**)

  - Inizializzazione contatore con il *valore di partenza*
  - **Ciclo Mentre** Contatore non ha raggiunto il *valore finale*
  - ...
  - Incremento contatore
  - **Fine-Mentre**

```
....
K = 1;
while ( K <= 10)
   {cout << K << endl;
    K++;
   }
……
```

  oppure

  - Inizializzazione contatore con il *valore di partenza*
  - **Ciclo Esegui**
  - ...
  - Incremento contatore
  - **Finché** Contatore non ha raggiunto il *valore finale*
  - ….

```
....
 K = 1;
 do {cout << K << endl;
      K++;
     } while ( K <= 10)
 ……
```