



### 3. Utilizzo degli output formattati

Il linguaggio C++ mette a disposizione manipolatori che consentono di rappresentare i risultati dell'elaborazione in modo più efficace.

La maggior parte di queste clausole è definibile con la funzione **setiosflags** (*imposta i flag per lo stream di input/output*) che imposta a 1 (vero) il bit di uno specifico formato. Per azzerare (falso) tale impostazione si deve richiamare la funzione **resetiosflags** (*azzerare i flag per lo stream di input/output*).

In generale, in informatica il termine **flag** indica un'impostazione che può assumere due valori opposti tra loro (come sì/no, vero/falso).

Entrambe le funzioni *setiosflags* e *resetiosflags* sono definite nel file di intestazione **iomanip** e prevedono un parametro di tipo *long*, detto **costante per la manipolazione dell'I/O**: la parola chiave che definisce il tipo di impostazione è preceduta dai caratteri **ios::** (*input/output stream*). Più precisamente, come vedremo in seguito, **ios** è la *classe base* per l'Input/Output e la coppia di caratteri **::**, come già visto in precedenza, si chiama *operatore di risoluzione di visibilità* (in inglese *scope resolution operator*), in questo caso *visibilità della classe*.

La tabella seguente riporta l'elenco dei principali valori previsti per le *costanti di manipolazione*.

Il termine **default** indica l'impostazione che viene assunta in *manca* di diversa dichiarazione. È possibile combinare più valori separandoli con l'operatore OR sui bit (**|**).

Costante	Descrizione
<b>ios::skipws</b>	elimina gli spazi in testa
<b>ios::left</b>	allinea a sinistra riempiendo i restanti caratteri a destra con il carattere di riempimento definito da <i>setfill()</i>
<b>ios::right</b>	allinea a destra ( <i>default</i> ) riempiendo i caratteri mancanti a sinistra con il carattere di riempimento definito da <i>setfill()</i>
<b>ios::internal</b>	inserisci i caratteri di riempimento solamente dopo l'eventuale segno e prima del valore
<b>ios::dec</b>	rappresenta i valori numerici in base 10 ( <i>default</i> )
<b>ios::oct</b>	rappresenta i valori numerici in base 8 (ottale)
<b>ios::hex</b>	rappresenta i valori numerici in base 16 (esadecimale)
<b>ios::showbase</b>	antepone la base ai numeri ottali (0) ed esadecimali (0x) secondo la notazione del linguaggio C++
<b>ios::showpoint</b>	mostra il punto decimale e i relativi zeri dei numeri in virgola mobile ( <i>float</i> e <i>double</i> ) che hanno valore intero (cioè non contengono decimali)
<b>ios::uppercase</b>	rappresenta le lettere dei numeri esadecimali e la lettera E (esponente) del formato scientifico in maiuscolo
<b>ios::showpos</b>	mostra il segno anche con i numeri positivi (+)
<b>ios::scientific</b>	rappresenta nel formato scientifico i numeri in virgola mobile ( <i>float</i> e <i>double</i> )
<b>ios::fixed</b>	rappresenta i numeri in virgola mobile ( <i>float</i> e <i>double</i> ) con un numero fisso di decimali

Per utilizzare le costanti occorre inserire all'inizio del programma la direttiva:

```
#include <iomanip>
```



Lo schema seguente illustra alcuni esempi di utilizzo di queste costanti:

Codice	Output
<pre>cout &lt;&lt; 1./3.   &lt;&lt; '\t'   &lt;&lt; setprecision(3)   &lt;&lt; 1./3.   &lt;&lt; '\t'   &lt;&lt; setiosflags(ios::scientific)   &lt;&lt; 1./3;</pre>	<pre>0.333333      0.333      3.333e-01</pre>
<pre>cout &lt;&lt; resetiosflags(ios::scientific)   &lt;&lt; endl;</pre>	<pre>(azzerata la notazione scientifica)</pre>
<pre>cout &lt;&lt; 10.   &lt;&lt; '\t'   &lt;&lt; setiosflags(ios::showpoint)   &lt;&lt; 10.   &lt;&lt; '\t'   &lt;&lt; setiosflags(ios::showpos)   &lt;&lt; setw(6) &lt;&lt; 10.   &lt;&lt; '\t'   &lt;&lt; setiosflags(ios::internal ios::showpos)   &lt;&lt; setw(6) &lt;&lt; 10.   &lt;&lt; resetiosflags(ios::internal ios::showpos)   &lt;&lt; '\ t'   &lt;&lt; setiosflags(ios::uppercase)   &lt;&lt; hex &lt;&lt; 10   &lt;&lt; dec   &lt;&lt; endl;  int valore1 = 100; int valore2 = 200; int totale;;</pre>	<pre>10      10.0      +10.0 + 10.0  A</pre>
<pre>cout &lt;&lt; setiosflags(ios::left)   &lt;&lt; setw(20) &lt;&lt; "Milano"   &lt;&lt; resetiosflags(ios::left)   &lt;&lt; setw(6) &lt;&lt; valore1   &lt;&lt; endl;  cout &lt;&lt; setiosflags(ios::left)   &lt;&lt; setw(20) &lt;&lt; "Roma"   &lt;&lt; resetiosflags(ios::left)   &lt;&lt; setw(6) &lt;&lt; valore2   &lt;&lt; endl;  totale = valore1 + valore2;  cout &lt;&lt; setfill('-')   &lt;&lt; setw(20) &lt;&lt; ' '   &lt;&lt; setw(6) &lt;&lt; '- '   &lt;&lt; setfill(' ')   &lt;&lt; endl;  cout &lt;&lt; setiosflags(ios::left)   &lt;&lt; setw(20) &lt;&lt; "Totale"   &lt;&lt; resetiosflags(ios::left)   &lt;&lt; setw(6) &lt;&lt; totale   &lt;&lt; endl;</pre>	<pre>Milano      100 Roma        200 ----- Totale      300</pre>



Si noti, in particolare, nel quarto esempio l'uso dell'impostazione **setiosflags (ios::left)** per l'allineamento a sinistra e di **resetiosflags(ios::left)** per l'annullamento di questa impostazione, ossia il ritorno all'allineamento a destra, che è l'impostazione di *default*.

Le operazioni di Input/Output utilizzano una zona di memoria temporanea, detta **buffer di I/O**, che contiene i dati che vengono ricevuti dal flusso (*stream*) di input o che devono essere inviati al flusso di output.

Il manipolatore **endl**, visto nei paragrafi precedenti, manda allo *stream* di output la sequenza di escape **'\n'** e svuota il buffer provocando l'uscita dei dati sul video.

Invece **'\n'** inserisce un ritorno a capo, ma non svuota il buffer.

Per forzare lo svuotamento del buffer si può usare il manipolatore **flush**, secondo la sintassi:

```
cout << flush;
```

Se si vuole rendere permanente per tutto il programma la richiesta di forzare lo *stream* di output a scaricare tutti i caratteri in coda in ciascuna operazione di scrittura, eliminando di fatto l'accodamento nel buffer, si deve utilizzare la costante **ios::unitbuf**:

```
cout << setiosflags(ios::unitbuf);
```

### PROGETTO • Calcolo del valore in dollari corrispondente al cambio di una cifra in euro.

Si deve costruire un programma che richieda all'utente l'importo in euro da cambiare e il cambio dollaro/euro, e che comunichi il corrispondente valore in dollari.

PROGRAMMA C++

```
// Cambio.cpp : cambio di denaro da euro a dollari
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    // input
    float euro, cambio;
    // output
    float dollari;

    cout << "Importo in euro: ";
    cin >> euro;
    cout << "Cambio dollari/euro: ";
    cin >> cambio;
    dollari = cambio * euro;

    cout << setiosflags(ios::left)
         << setw(12) << "Euro"
         << setw(12) << "Cambio"
         << setw(12) << "Dollari"
         << endl;
```



```
cout << setiosflags(ios::left)
      << setiosflags(ios::fixed)
      << setprecision(2)
      << setw(12) << euro
      << setprecision(4)
      << setw(12) << cambio
      << setprecision(2)
      << setw(12) << dollari
      << endl;

return 0;
}
```

Il programma mostra esempi di utilizzo degli output formattati per visualizzare in modo più ordinato i risultati dell'elaborazione. Le variabili utilizzate sono state dichiarate di tipo *float*, in quanto sia i dollari sia gli euro sono importi con i decimali e perché il valore del cambio può avere una parte decimale.

## ESERCIZI

1. Scrivere un programma che legga due numeri interi e che comunichi il risultato della divisione tra i due numeri presentando 3 cifre intere e 5 decimali.
2. Dato il raggio, calcolare la circonferenza e l'area del cerchio. Presentare i risultati con 3 cifre decimali.
3. Dati in input la descrizione, la quantità, il prezzo unitario di un articolo venduto e l'aliquota IVA, comunicare in output la descrizione e il prezzo totale aumentato dell'IVA, visualizzando i dati in modo formattato.
4. Scrivere un programma che richieda in input il nome dell'utente e scriva sul video il messaggio «Buongiorno,» seguito dal nome fornito.