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** (*azzera 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 *setioflags* e *resetioflags* 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 *mancanza* di diversa dichiarazione.
È possibile combinare più valori separandoli con l'operatore OR sui bit (|).

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

`cout << 1./3.`
`<< '\t'`
`<< setprecision(3)`
`<< 1./3.`
`<< '\t'`
`<< setiosflags(ios::scientific)`
`<< 1./3;`
	`0.333333  0.333  3.333e-01`

`cout << resetiosflags(ios::scientific)` `<< endl;`	(azzera la notazione scientifica)

`cout << 10.`
`<< '\t'`
`<< setiosflags(ios::showpoint)`
`<< 10.`
`<< '\t'`
`<< setiosflags(ios::showpos)`
`<< setw(6) << 10.`
`<< '\t'`
`<< setiosflags(ios::internal|ios::showpos)`
`<< setw(6) << 10.`
`<< resetiosflags(ios::internal|ios::showpos)`
`<< '\ t'`
`<< setiosflags(ios::uppercase)`
`<< hex << 10`
`<< dec`
`<< endl;`
 	
`int valore1 = 100;`
`int valore2 = 200;`
`int totale;;`
	`10  10.0  +10.0  + 10.0  A`

`cout << setiosflags(ios::left)`
`<< setw(20) << "Milano"`
`<< resetiosflags(ios::left)`
`<< setw(6) << valore1`
`<< endl;`
 	
`cout << setiosflags(ios::left)`
`<< setw(20) << "Roma"`
`<< resetiosflags(ios::left)`
`<< setw(6) << valore2`
`<< endl;`
 	
`totale = valore1 + valore2;`
 	
`cout << setfill('-')`
`<< setw(20) << ' '`
`<< setw(6) << '-'`
`<< setfill(' ')`
`<< endl;`
 	
`cout << setiosflags(ios::left)`
`<< setw(20) << "Totale"`
`<< resetiosflags(ios::left)`
`<< setw(6) << totale`
`<< endl;`
	`Milano  100`
	`Roma  200`
	`--------------------  ------`
	`Totale  300`

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.