Format Specifier

Since we're tackling text, it would only be appropriate to have tools which can help us format our our data.

Format specifiers enable you to adjust the input and output data explicitly.

> **i I use manipulators as format specifiers**
> The format specifiers are available as manipulators and flags. I only present manipulators in this book because their functionality is quite similar and manipulators are more comfortable to use.

```cpp
#include <iostream>

int main(){

  std::cout << std::endl;

  int num{2011};

  std::cout << num << "\n\n";

  std::cout.setf(std::ios::hex, std::ios::basefield);
  std::cout << num << std::endl;
  std::cout.setf(std::ios::dec, std::ios::basefield);
  std::cout << num << std::endl;

  std::cout << std::endl;

  std::cout << std::hex << num << std::endl;
  std::cout << std::dec << num << std::endl;

  std::cout << std::endl;

}
```

Manipulators as format specifiers

The followings tables present the important format specifiers. The format specifiers are sticky except for the field width, which is reset after each application.

The manipulators without any arguments require the header `<iostream>`, and the manipulators with arguments require the header `<iomanip>`.

Manipulator	Stream type	Description
`std::boolalpha`	input and output	Displays the boolean as a word.
`std::noboolalpha`	input and output	Displays the boolean as number (default).

Displaying of boolean values

Manipulator	Stream type	Description
`std::setw(val)`	input and output	Sets the field width to `val`.
`std::setfill(c)`	output stream	Sets the fill character to `c` (default: spaces).

Set the field width and the fill character

Manipulator	Stream type	Description
`std::left`	output	Aligns the output left.
`std::right`	output	Aligns the output

		right.
`std::internal`	output	Aligns the signs of numbers left, the values right.

Alignment of the text

Manipulator	Stream type	Description
`std::showpos`	output	Displays positive signs.
`std::noshowpos`	output	Doesn’t display positive signs (default).
`std::uppercase`	output	Uses upper case characters for numeric values (default).
`std::lowercase`	output	Uses lower case characters for numeric values.

Positive signs and upper or lower case

Manipulator	Stream type	Description
`std::oct`	input and output	Uses natural numbers

`std::oct`	input and output	in octal format.
`std::dec`	input and output	Uses natural numbers in decimal format (default).
`std::hex`	input and output	Uses natural numbers in hexadecimal format.
`std::showbase`	output	Displays the numeric base.
`std::noshowbase`	output	Doesn’t display the numeric base (default).

Display of the numeric base

There are special rules for floating point numbers:

- The number of significant digits (digits after the comma) is by default 6.
- If the number of significant digits is not big enough the numbers is displayed in scientific notation.
- Leading and trailing zeros are not be displayed.
- If possible the decimal point are not be displayed.

Manipulator	Stream type	Description
`std::setprecision(val)`	output	Adjusts the precision of the output to `val`.
`std::showpoint`	output	Displays the decimal point.

`std::noshowpoint`	output	Doesn't display the decimal point (default).
`std::fixed`	output	Displays the floating point number in decimal format.
`std::scientific`	output	Displays the floating point number in scientific format.
`std::hexfloat`	output	Displays the floating-point number in hexadecimal format.
`std::defaultfloat`	output	Displays the floating-point number in default floating-point notation.

Floating point numbers

```
#include <iomanip>
#include <iostream>

int main(){

  std::cout << std::endl;

  std::cout << "std::setw, std::setfill and std::left, right and internal: " << std::endl;

  std::cout.fill('#');
  std::cout << -12345 << std::endl;
  std::cout << std::setw(10) << -12345 << std::endl;
  std::cout << std::setw(10) << std::left << -12345 << std::endl;
  std::cout << std::setw(10) << std::right << -12345 << std::endl;
  std::cout << std::setw(10) << std::internal << -12345 << std::endl;

  std::cout << std::endl;

  std::cout << "std::showpos:" << std::endl;

  std::cout << 2011 << std::endl;
  std::cout << 2011 << std::endl;
  std::cout << std::showpos << 2011 << std::endl;



  std::cout << std::noshowpos << std::endl;

  std::cout << "std::uppercase: "  << std::endl;
  std::cout << 12345678.9 << std::endl;
  std::cout << std::uppercase << 12345678.9 << std::endl;

  std::cout << std::nouppercase << std::endl;

  std::cout << "std::showbase and std::oct, dec and hex: " << std::endl;
  std::cout << 2011 << std::endl;
  std::cout << std::oct << 2011 << std::endl;
  std::cout << std::hex << 2011 << std::endl;

  std::cout << std::endl;

  std::cout << std::showbase;
  std::cout << std::dec << 2011 << std::endl;
  std::cout << std::oct << 2011 << std::endl;
  std::cout << std::hex << 2011 << std::endl;

  std::cout << std::dec << std::endl;

  std::cout << "std::setprecision, std::fixed and std::scientific: " << std::endl;

  std::cout << 123.456789 << std::endl;
  std::cout << std::fixed << std::endl;
  std::cout << std::setprecision(3) << 123.456789 << std::endl;
  std::cout << std::setprecision(4) << 123.456789 << std::endl;
  std::cout << std::setprecision(5) << 123.456789 << std::endl;
  std::cout << std::setprecision(6) << 123.456789 << std::endl;
  std::cout << std::setprecision(7) << 123.456789 << std::endl;
  std::cout << std::setprecision(8) << 123.456789 << std::endl;
  std::cout << std::setprecision(9) << 123.456789 << std::endl;

  std::cout << std::endl;
  std::cout << std::setprecision(6) << 123.456789 << std::endl;
  std::cout << std::scientific << std::endl;
  std::cout << std::setprecision(6) << 123.456789 << std::endl;
  std::cout << std::setprecision(3) << 123.456789 << std::endl;
  std::cout << std::setprecision(4) << 123.456789 << std::endl;
  std::cout << std::setprecision(5) << 123.456789 << std::endl;
  std::cout << std::setprecision(6) << 123.456789 << std::endl;
  std::cout << std::setprecision(7) << 123.456789 << std::endl;
  std::cout << std::setprecision(8) << 123.456789 << std::endl;
  std::cout << std::setprecision(9) << 123.456789 << std::endl;

  std::cout << std::endl;

}
```

Format specifier