

Format Specifier

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

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

i We use manipulators as format specifiers

The format specifiers are available as manipulators and flags. We only present manipulators in this course because their functionality is quite similar and manipulators are more comfortable to use.

```
#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 following tables present the important format specifiers. They 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
<code>std::boolalpha</code>	input and output	Displays the boolean as a word.
<code>std::noboolalpha</code>	input and output	Displays the boolean as number (default).

Displaying boolean values

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

Set the field width and the fill character

Manipulator	Stream type	Description
<code>std::left</code>	output	Aligns the output left.
<code>std::right</code>	output	Aligns the output right.

<code>std::internal</code>	output	Aligns the signs of numbers left, the values right.
----------------------------	--------	---

Alignment of the text

Manipulator	Stream type	Description
<code>std::showpos</code>	output	Displays positive signs.
<code>std::noshowpos</code>	output	Doesn't display positive signs (default).
<code>std::uppercase</code>	output	Uses uppercase characters for numeric values (default).
<code>std::lowercase</code>	output	Uses lower case characters for numeric values.

Positive signs and upper or lower case

Manipulator	Stream type	Description
<code>std::oct</code>	input and output	Uses natural numbers

<code>std::oct</code>	input and output	in octal format.
<code>std::dec</code>	input and output	Uses natural numbers in decimal format (default).
<code>std::hex</code>	input and output	Uses natural numbers in hexadecimal format.
<code>std::showbase</code>	output	Displays the numeric base.
<code>std::noshowbase</code>	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 are displayed in scientific notation.
- Leading and trailing zeros are not displayed.
- If possible, the decimal points are not displayed.

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

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

Floating point numbers

There's a try it out exercise for us in the next lesson.