sign up log in tour help

How do I turn off std::fixed and return to the default C++ setting

My code:

```
std::vector<double> thePoint(4);
thePoint[0] = 86;
thePoint[1] = -334.8836574;
thePoint[2] = 24.283;
thePoint[3] = 345.67675;

ofstream file1(tempFileName, ios::trunc);
file1 << std::setprecision(16)              << thePoint[0] << " ";
file1 << std::fixed << std::setprecision(2) << thePoint[1] << " ";
file1 << std::setprecision(16)              << thePoint[2] << " ";
file1 << std::setprecision(16)              << thePoint[3];
```

I get:

86 -334.88 24.28300000000000 345.67675000000000

I want:

86 -334.88 24.283 345.67675

The odd formatting is needed for an interface with other picky code.

c++

edited Aug 23 '12 at 15:26

asked Aug 23 '12 at 15:17
user1247549
83 1 8

I always wondered that whenever I read a C++ book (never actually used it myself yet)... – Mehrdad Aug 23 '12 at 15:18

If you want just 2 decimal places, why do you request 16 places by using `std:setprecision(16)` ? – Jan Spurny Aug 23 '12 at 15:20

Anyway I don't understand what you actualy want - in the title you say you want to "turn off std::fixed" (you can do that by using std::scientific) and then you say you want just two decimal places in output (which you'll get if you just use the right precision, i.e. 2 instead of 16). – Jan Spurny Aug 23 '12 at 15:25

2 Answers

You should do this:

```
file1 << std::fixed << std::setprecision(2) << thePoint[1] << " ";
file1.unsetf(ios_base::fixed);
file1 << std::setprecision(16)              << thePoint[2];
```

The `floatfield` format flag can take any of its two possible values (using the manipulators `fixed` and `scientific`), or none of them (using `ios_base::unsetf`).

answered Aug 23 '12 at 15:20
Andrey
5,608 3 15 43

You can do it by forcing the floatfield to an empty value:

```
file1.setf( std::ios_base::fmtflags(), std::floatfield );
```

In practice, it's rare to want to, however. The usual protocol is to save the format flags, and restore them when you're through:

```
std::ios_base::fmtflags originalFlags = file1.flags();
//  ...
file1.flags( originalFlags );
```

Of course, you'd normally use RAII to do this in a real program. You should have a `IOSave` class in your toolbox which will save the flags, the precision and the fill character in its constructor, and restore them in the destructor.

It's also not very good practice to use `std::setprection` etc. directly. A better solution would be to define your own manipulators, with names like `pression` or `volume`, and use those. This is logical markup, and means that you control the format for e.g. pression from one central location, rather than having it spread throughout the program. And if you write your own manipulators, it's relatively easy to have them restore the original formatting parameters at the end of the full expression. (The manipulator objects will be temporaries, destructed at the end of the full expression.)

answered Aug 23 '12 at 16:32

James Kanze
116k 7 95 223