Montreal Code Sprint

LIBPC – IMPLEMENTATION

mpg
Flaxen Geo
15 March 2011

Contents

(nope)

Source Layout

- apps/ -- source for the CL apps
- bin/ -- generated libs and exes
- cmake/ -- (internal)
- csharp/ -- mpg sandbox
- doc/ -- documentation
- include/ -- all public headers
- src/ -- lib's sources, private headers
- test/ -- unit tests, data files

Development Requirements

- Same as libLAS
  - Python, CMake, …
  - but, all of Boost
- Useful:
  - lastools, laszip, liblas, gdal
  - RST tools
  - …

Documentation

- doc/notes
  - rst-formatted Notes files
  - capturing design notes, styles, conventions, etc
  - future website fodder
- website
  - **to be resolved**
- header files
  - **to be resolved**

Boost

- Use Boost libraries whenever possible
- Currently
  - program_options
  - cstdint.hpp (boost::uint32_t)
  - uuid class
  - I/O streams
  - unit test framework
  - …?

Unit Tests

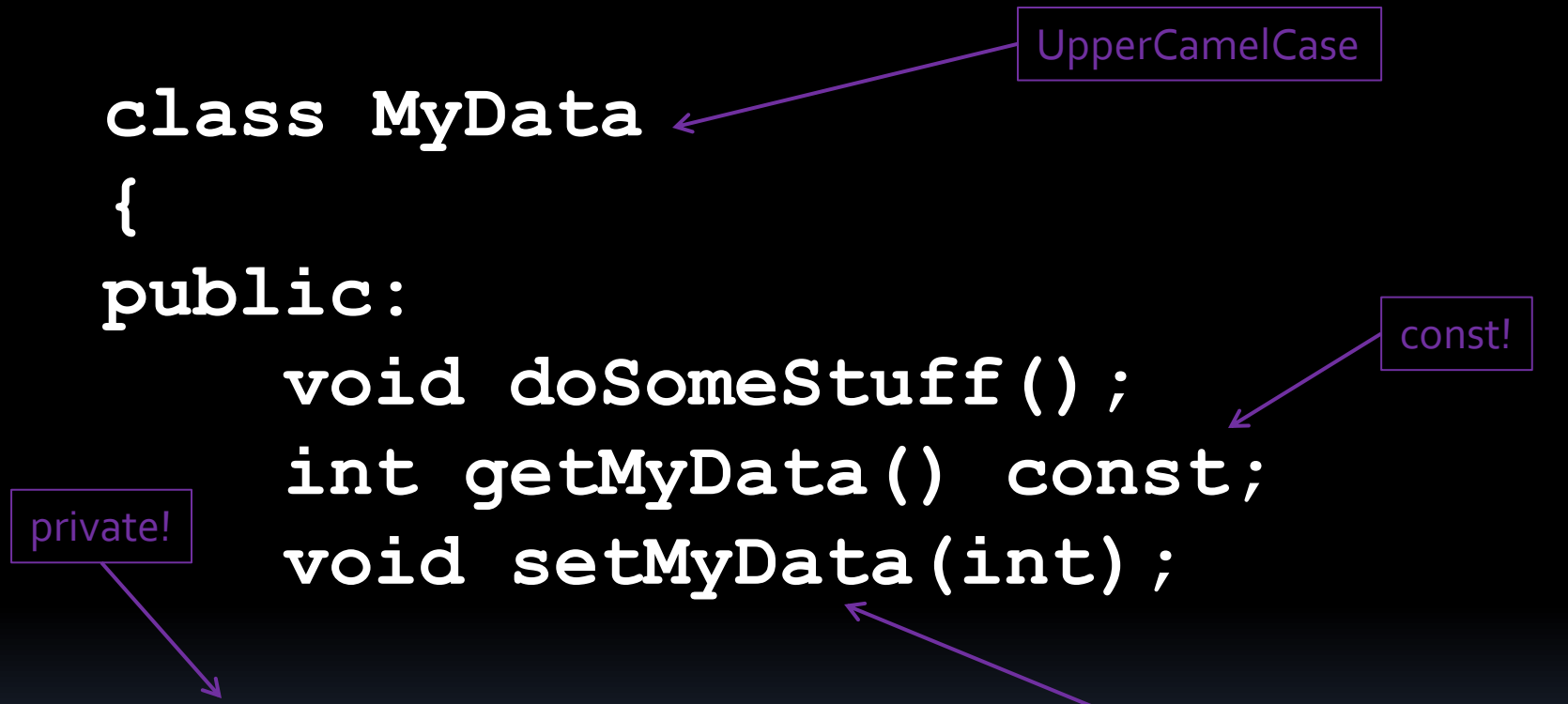
- sources in test/unit
  - files named <CLASS>Test.cpp
  - cut-n-paste to make new tests
- Goal: every class has a Test file
  - even if all it does is verify the ctor works
- binary goes into bin/
- assume working dir is where the .exe is
- reference files in test/data
- **Need to add tests for CL apps**

Coding Style (1)

- astyle tool
- 4 spaces, no tabs, unix line feeds
- Curly brackets are done the only sensible way

```
if (p)
{
    foo();
}
else
{
    bar();
}
```

Coding Style (2)




```
class MyData
{
public:
    void doSomeStuff();
    int getMyData() const;
    void setMyData(int);

private:
    int m_myData;
};
```

Coding Style (3)

- Extensions: .cpp, .hpp
- File names to match the class names
  - **"reader" issue to be resolved**

Command-line Apps

- Uses boost::program_options
- Simple framework for common look/feel
- **Usage, Help, Errors to stdout or stderr?**

CMake & Macros

- libpc_defines.h /* generated C source! */
- Contains
  - 3rd party lib info
    - LIBPC_HAVE_GDAL, …
  - version info
    - LIBPC_VERSION_MAJOR, …
  - build info
    - LIBPC_BUILD_TYPE ("Debug", etc)
- libpc_config.hpp contains functions for these

- **Platform macros? LIBPC_OS_WIN32, ..?**

export.hpp

- Just defines the LIBPC_DLL macro
  - declspec(dllexport)

libpc.hpp?

- Do we want a single header for public users?
  - forward declarations
  - #include libpc_defines.h, etc
  - #includes for common libpc classes

Dumps

- Goal: a dump routine for every class

```
std::ostream& operator<<(std::ostream&,
                         const MyClass&);

// useful for in debugger
void MyClass::dump() const
{ std::cout << *this; }
```

SWIG and C#

- Try to avoid things that cause swig-pain
  - nontrivially templated classes
  - use of multiple inheritance
  - overly complicated pointer'd parameters
- We have C# bindings now
  - (proof-of-concept, mpg-only)
  - and will want Python later
  - **needs to be built by CMake**

The Big Renaming

- Need to set up driver directories
  - Classes
    - libpc::LiblasReader → libpc::liblas::Reader ???
  - Files
    - include/libpc/LiblasReader.hpp → include/libpc/drivers/liblas/Reader.hpp ???
  - Need also to fix CMake for VS IDE project file folder happiness

Warnings

- Treat warnings as errors

Copy ctor & operator==

- Always declare both
  - Avoid compiler-provided implementation
- Implement it
  - or mark the declaration as "// not implemented"

Miscellany

- typedef boost::uint32_t uint32_t ?
- strings – need to do something about exception mssg strings?