Stephan Aiche
AG Algorithmische Bioinformatik
Fachbereich Mathematik und Informatik

Freie Universität Berlin

Coding Convention

Allgemein

Eine aktuelle Version findet man hier

http://www-bs2.informatik.uni-tuebingen.de/services/OpenMS/OpenMS-internal/coding_conventions.html

Formatierung

- 2 Leerzeichen Tab breite
- Klammern von der Operation Trennen

```
while (continue == true)
{
  for (int i = 0; i < 10; i++)
  {
    ...
  }

  if (x < 7)
  {
    ....
  }
}
```

Datei-Header und Guards

```
// -*- mode: C++; tab-width: 2; -*-
// vi: set ts=2:
//
... copyright header, not shown ...
//
// --------------------------------------------------------------------------
// $Maintainer: Heinz Erhardt $
// --------------------------------------------------------------------------

#ifndef OPENMS_KERNEL_DPEAK_H
#define OPENMS_KERNEL_DPEAK_H

#include <OpenMS/CONCEPT/Types.h>

#include <functional>
#include <sstream>

namespace OpenMS
{
   ... the actual code goes here ...
} // namespace OpenMS

#endif // OPENMS_KERNEL_DPEAK_H
```

Klassen

```
class OPENMS_DLLAPI Test
{
  public:

    // default constructor
    Test();

    // copy constructor
    Test(const Test& test);

    // destructor
    virtual ~Test();

    // assignment operator
    Test& operator = (const Test& test)
    {
      //ALWAYS CHECK FOR SELF ASSIGNEMT!
      if (this == &test) return *this;
      //...
      return *this;
    }
};
```

Primitive Datentypen

OpenMS definiert seine eigenen Typen in

`OpenMS/include/OpenMS/CONCEPT/Types.h`

z.B.

- `DoubleReal`
- `Int`
- `Size`
- `SignedSize`
- `...`

Namespaces

- Wenn euer Code direkt in OpenMS eingebaut wird

```
namespace OpenMS
```

- Wenn euer Code im externen Projekt liegt

```
namespace OpenMSExternal
```

- Niemals im Header

```
using namespace std;
```

Freie Universität Berlin

Getter / Setter

```
class Test {
public:
  const vector<String>& getMember() const
  {
   return member_;
  }
  void setMember(const vector<String>& name)
  {
    name_ = member_;
  }

  vector<String>& getMember()
  {
    return member_;
  }
};
```

Exception handling

- Alle Exception Klassen sind abgeleitet von

```
Exception::Base
```

- Throwing exceptions

```
throw AnyException(__FILE__, __LINE__, __PRETTY_FUNCTION__);
```

•Catching exceptions

```
try
{
   // some code which might throw
}
catch ( Exception& e)
{
   // Handle the exception, then possibly re-throw it:
   // throw; // the modified e
}
```

Exceptions II

Exceptions spezifizieren

```
/**
  @brief Silly function

  @exception Exception::Foo is always thrown
*/
void myFunction() {
  throw Foo(__FILE__, __LINE__, __PRETTY_FUNCTION__);
}
```

Naming conventions

- File names

```
ClassName.(h|C)
```

- Underscores
  - Teile der Namen werden durch _ getrennt
  - _ am Ende kennzeichnen `private` oder `protected` Member/Klassen
- Class / type / namespace names

```
class Simple; //ordinary class
class SimpleThing; //ordinary class
class PDBFile; //using an abbreviation
class Buffer_; //protected or private nested
class class ForwardIteratorTraits_; // protected or private
                                    // nested class
```

Naming conventions II

•Variable names

```
int simple; //ordinary variable
bool is_found; //ordinary variable
string MS_instrument; //using an abbreviation
int counter_; //protected or private member
int persistent_id_; //protected or private member
```

• Function names/method names

```
//ordinary function, no arguments
void hello();
//ordinary function
int countPeaks(PeakArray const& p);
//ordinary function with an unused argument
bool ignore(string& /* name */);
//an ordinary function
bool isAdjacentTo(Peak const * const * const & p) const;
//protected or private member function
bool doSomething_(int i, string& name);
```

Naming conventions III

•Enums and preprocessor constants

```
#define MYCLASS_SUPPORTS_MIN_MAX 0 //preprocessor constant
enum DimensionId { DIM_MZ = 0, DIM_RT = 1 }; //enumerated
values
enum DimensionId_ { MZ = 0, RT = 1 }; //enumerated values
```

- Parameter
  - nur kleine Buchstaben und _
  - Für numerische Werte sollten man den Wertebereich mit angeben
  - Wenn möglich sollten Einheiten in der Beschreibung angegeben werden

Documentation

Testing

Macros to start, finish and evaluate tests

START_TEST(class_name, version)
Start of a class test file (initialization)

END_TEST()
End of a class test file (cleanup)

START_SECTION(name)
Start of a method test. If the name starts with '[EXTRA]' it does not have to match a methods name.

END_SECTION()
End of a single test

STATUS(message)
Shows a status message e.g. used to show the progress of a test preparations that take a while

ABORT_IF(condition)
Skip remainder of substest if condition holds

Comparison macros

TEST_EQUAL(a, b)
Tests if two expressions are equal

TEST_NOT_EQUAL(a, b)
Tests if two expressions are not equal

TEST_REAL_SIMILAR(a, b)
Tests if two real numbers are equal (within a margin)

TEST_STRING_EQUAL(a, b)
Tests if *a* and *b* are equal as strings

TEST_STRING_SIMILAR(a, b)
Tests if *a* and *b* are similar as strings - allowing numerical deviations and differing whitespaces

TOLERANCE_ABSOLUTE(double)
Sets the absolute difference allowed when testing floating point numbers

TOLERANCE_RELATIVE(double)
Sets the relative difference allowed when testing floating point numbers

TEST_EXCEPTION(exception, expression)
Tests if the expression throws the exception

TEST_EXCEPTION_WITH_MESSAGE(exception, expression, message)
Tests if the expression throws the exception and if the exception has the message

TEST_FILE_EQUAL(file, template_file)
Tests if two files are identical

TEST_FILE_SIMILAR(file, template_file)
Tests if two files are similar - allowing numerical deviations and differing whitespaces

Temporary files

You might want to create temporary files during the tests. The following macro puts a temporary filename into the string argument. The file is automatically deleted after the test.

All temporary files are validated using the XML schema,if the type of file can be determined by FileHandler. Therefor for each file written in a test NEW_TMP_FILE should be called. Otherwise only the last writen file is checked.

NEW_TMP_FILE(string)

Vermeidet

```
if (isValid(a))
  return 0;
```