

2/25/2010


MITK Style Guide


Beautiful Code



GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

- Using case change to indicate separate words

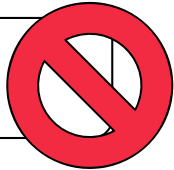
Imagefilter
Pixeltype
Datastorage 

ImageFilter
PixelFormat
DataStorage 

- Underscores are not used

Image_Filter
_Node 

- Variable names should convey the meaning behind the code

`BoundingBox::Pointer result = BoundingBox::New();` 

`BoundingBox::Pointer boundingBox = BoundingBox::New();` 

- Names are generally spelled out

`mitk::DataTreeNode* n;`



`mitk::DataTreeNode* node;`



- Abbreviation are allowable when in common use

`ROI`



- Classes are named beginning with a capital letter
- Classes are named according to the following general rule:

`class name = <algorithm><input><concept>`

Examples of concepts:

Container	A container of objects such as points or images	<i>VectorContainer</i>
Factory	Object factories are used to create instances	<i>DataTreeNodeFactory</i>
Filter	A class that participates in the data processing pipeline	<i>AddImageFilter</i>
Mapper	Transform data from one form into another	<i>Contour2DMapper</i>
Reader	A class that reads a single data object	<i>VtkSurfaceReader</i>

- Files should have the same name as the class, with an `mitk` Qt specific MITK classes with an `Qmitk` prepended.

`mitkDataStorage`
`QmitkDataStorageComboBox`



- Header files ends with an `.h`
- Implementation files with an `.cpp` or `.txx` for a template class

- Functions and methods are named beginning with a capital letter
- Referring to class methods in code, an explicit `this->` pointer should be used

```
mitk::DataStorage::SetOfObjects::ConstPointer all = GetAll();
```



```
mitk::DataStorage::SetOfObjects::ConstPointer all = this->GetAll();
```



Slots

- Slots are named according to the following general rule:

```
On[variable name who send the signal][signal]();
```

```
connect( loadImagePushButton, SIGNAL( clicked( bool ) ),  
        SLOT( OnLoadImagePushButtonClicked( bool ) ) );
```

```
void mitk::Image::OnLoadImagePushButtonClicked( bool )  
{  
    ... Do something ...  
}
```

Signals

- Signals are named according to the following general rule

```
Signal[MethodName](); e.g. emit SignalPointListChanged();
```

Class Data Members

- Class data member are prepended with m_
- Except of QT class Data Members, those begins in lowercase.

m_Volumes
m_OffsetTable
m_ImageMask



loadImageButton
closeImageAction



Local Variables

- Local variables begin in lowercase

offset
data
slicesIt



Qt Variables

- GUI variables ends with name of used QT tool.

QPushButton* loadImagePushButton;
QAction* closeImageAction;
QCheckBox* hideImageCheckBox;
QRadioButton* binaryImageRadioButton;



Typedefs

- Typedef names end in the word `Type`


```
typedef TPixel  
typedef itk::Image< TPixel, VImageDimension >  
typedef std::list<mitk::Image::Pointer>
```

PixelType;
ImageType;
ImageListType;




Declaration of Pointers


- Position of * pointers are connected with declaration type

`int *counter;` 

`int* counter;` 


- Analog to references:

`int &counter;` 


`int& counter;` 

- MITK classes should be in namespace mitk

```
mitk::Image::Pointer mitk::ImageGenerator::MakeImage()  
{  
    mitk::Image::Pointer image = mitk::Image::New();  
    mitk::ImageDecorator::Pointer decorator = mitk::ImageDecorator::New();  
    decorator ->Decorate( image );  
    return image;  
}
```

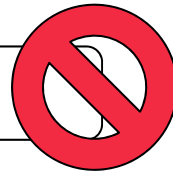


```
mitk::Image::Pointer mitk::ImageGenerator::MakeImage()  
{  
    // already in namespace mitk here!  
    Image::Pointer image = Image::New();  
    ImageDecorator::Pointer decorator = ImageDecorator::New();  
    decorator ->Decorate( image );  
    return image;  
}
```



- Each line of code should take no more than 120 characters.
- Use lots of whitespace to separate logical blocks of code, intermixed with Comments
- **DO NOT USE TABS.** Set up your editor to insert 2 spaces.
- Declaration of variables should be one declaration per line

```
int counter, imageNumber;
```



```
int counter;  
int imageNumber;
```



```
/*=====
Program: Medical Imaging & Interaction Toolkit
Module: $RCSfile$
Language: C++
Date: $Date: $
Version: $Revision: $
Copyright (c) German Cancer Research Center, Division of Medical and
Biological Informatics. All rights reserved.
See MITKCopyright.txt or http://www.mitk.org/copyright.html for details.
This software is distributed WITHOUT ANY WARRANTY; without even
the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the above copyright notices for more information.
*/
```

Copyright

```
#ifndef mitkClassName_h
#define mitkClassName_h
```

#include Guards

```
#include "... .h"
```

Includes [A...Z]

```
/*!
\brief mitkClassName
\sa .....
\verbatim
Last contributor: $Author: $
\endverbatim
*/
```

Doxygen

```
namespace mitk
```

Namespace

```
{  
  template <class TType>  
  class ClassName : public ImageBase<VImageDimension>  
  {
```

Class (Template)

```
public:
```

```
  ....typedefs....
```

Typedefs

```
public:
```

```
  ....methods....
```

```
protected:
```

```
  ....methods....
```

Methods

```
private:
```

```
  ....methods....
```

```
signals:
```

```
  Signal...();
```

QT Signals

```
public slots:
```

```
  On...();
```

```
protected slots:
```

```
  On...();
```

QT Slots

```
private/protected:
```

```
  ....class data members....
```

Data Members

```
};
```


```
}
```

```
#endif // mitkClassName_h
```

#include Guards


- Used to delimit the scope of an if, for, while, switch.
- Braces are placed on a line by themselves

```
for ( unsigned int i = 0; i < 3; ++i )  
{  
    ... do something ...  
}
```




- You can choose to use braces on a line with a code block when the block consists of a single line

```
if ( condition ) { foo = 1; }  
else if ( condition2 ) { foo = 3; }  
else { return; }  
  
for ( unsigned int i = 0; i < 3; ++i ) { x[i] = 0.0; }
```



```
if ( condition )  
{  
    ... do something ...  
}  
else if ( other condition )  
{  
    ... do something ...  
}  
else  
{  
    ... do something ...  
}
```



- `#include` guard is a particular construct used to avoid the problem of double inclusion when dealing with the `#include` directive.

ClassName_h

File: grandfather.h

```
struct foo  
{  
    int m_Member;  
};
```

File: father.h

```
#include "grandfather.h"
```

File: child.h

```
#include "grandfahther.h"  
#include "father.h"
```



File: grandfather.h

```
#ifndef grandfather_h  
#define grandfather_h  
struct foo  
{  
    int m_Member;  
};  
#endif
```

File: father.h

```
#include "grandfather.h"
```

File: child.h

```
#include "grandfahther.h"  
#include "father.h"
```



- KWStyle is integrated in the software process to ensure that the code written by several users is consistent and can be viewed/printed as it was written by one person.
- KWStyle is primarily checking C/C++ source code and it assumes that the code is syntactically correct, i.e it compiles on a standard compiler.
- Among the features provided by KWStyle:
 - Several Indentation checking
 - Copyright Header correctness
 - Maximum line length
 - Internal variable checking via regular expressions
 - New line at the end of file

- KWStyle can be integrated with Dart as part of the dashboard
(Nightly Checked Projects: ITK, VTK, Cmake, IGStk)

Filename	LEN	IVR	IVA	SEM	DCL	EOF	TAB	ESP	IND	HRD	DEF	TDR	TDA	NMS	NMC	WCM	MCM	EML	TPL	OPS
/projects/KWStyle/Insight/Code/Common																				
itkBSplineDeformableTransform.h	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
itkChainCodePath2D.h	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
itkConceptChecking.h	0	0	0	0	0	0	0	0	57	0	0	0	6	0	0	0	0	0	8	0
itkConstShapedNeighborhoodIterator.h	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
itkCoreAtomImageToDistanceMatrixProcess.txx	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
itkCovarianceImageFunction.txx	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
itkCovariantVector.cxx	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3	0	0
itkCovariantVector.txx	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	3	0	0
itkDataObject.h	0	0	0	0	0	0	0	0	2	0	0	0	2	0	0	0	2	0	0	0
itkDataObjectDecorator.h	0	0	0	0	0	1	0	0	0	0	0	0	4	0	0	0	0	0	0	0
itkDataObjectDecorator.txx	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
itkDecisionRuleBase.cxx	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
itkDecisionRuleBase.h	0	0	0	2	0	1	0	0	0	0	0	0	3	0	0	0	0	0	0	0
itkDefaultDynamicMeshTraits.h	0	0	0	0	0	0	0	0	4	0	0	0	2	0	0	0	0	0	0	0
itkDefaultImageTraits.h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
itkDefaultImageAccessor.h	0	0	0	2	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
itkDefaultPixelAccessorFunction.h	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
itkDefaultStaticMeshTraits.h	0	0	0	0	0	0	0	0	3	0	0	0	4	0	0	0	0	0	0	0
itkDefaultVectorPixelAccessor.h	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
itkDefaultVectorPixelAccessorFunction.h	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
itkDenseFiniteDifferenceImageFilter.h	0	0	0	0	0	0	0	0	2	0	1	0	3	0	0	0	0	0	0	0
itkDenseFiniteDifferenceImageFilter.txx	0	0	0	0	0	0	0	1	0	0	1	0	4	0	0	0	0	0	0	0
itkDerivativeOperator.h	0	0	0	0	0	1	0	0	2	0	0	0	1	0	0	0	0	0	0	0