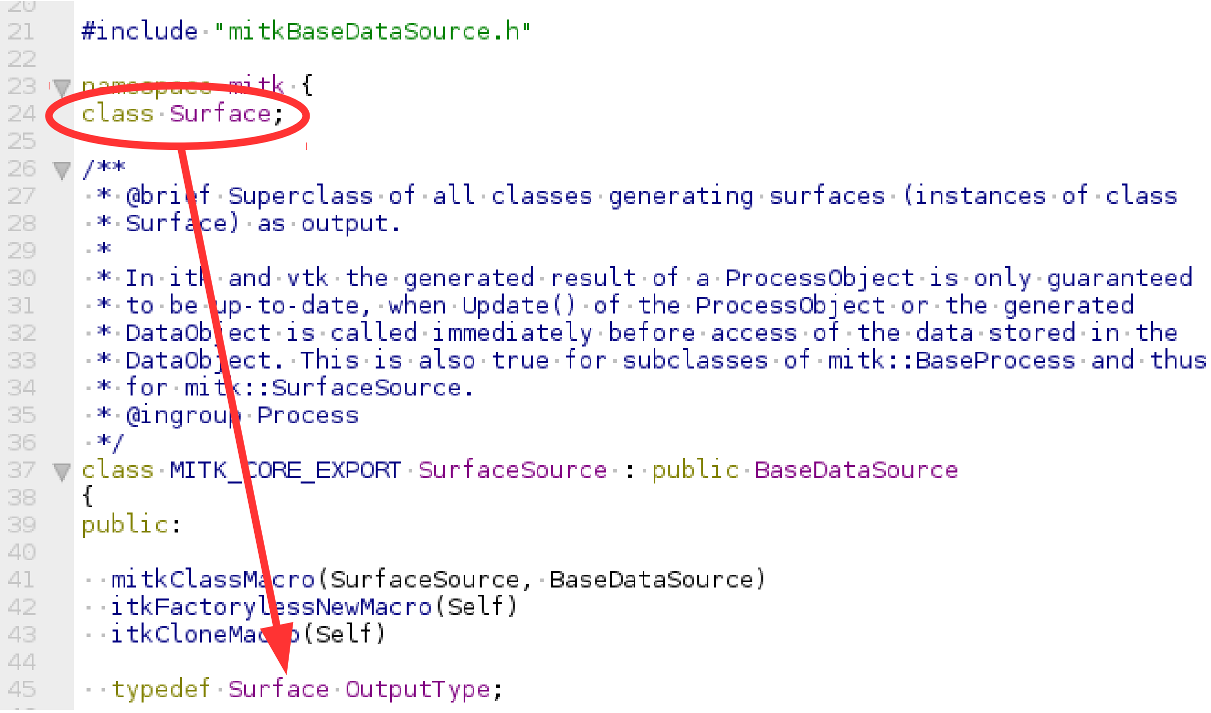
28.01.15

Forward Declaration

Bugsquashing Seminar
Nico Riecker

Forward declaration - Whats this ?!

- Declaration of an identifier (variable, function, class ...)
- Makes the identifier available in your class with some restrictions
- Also called incomplete type – Compiler only knows „It exists“



```
#include "mitkBaseDataSource.h"

namespace mitk {
class Surface;

/**
 * @brief Superclass of all classes generating surfaces (instances of class
 * Surface) as output.
 *
 * In itk and vtk the generated result of a ProcessObject is only guaranteed
 * to be up-to-date, when Update() of the ProcessObject or the generated
 * DataObject is called immediately before access of the data stored in the
 * DataObject. This is also true for subclasses of mitk::BaseProcess and thus
 * for mitk::SurfaceSource.
 * @ingroup Process
 */
class MITK_CORE_EXPORT SurfaceSource : public BaseDataSource
{
public:

  mitkClassMacro(SurfaceSource, BaseDataSource)
  itkFactorylessNewMacro(Self)
  itkCloneMacro(Self)

  typedef Surface OutputType;
```

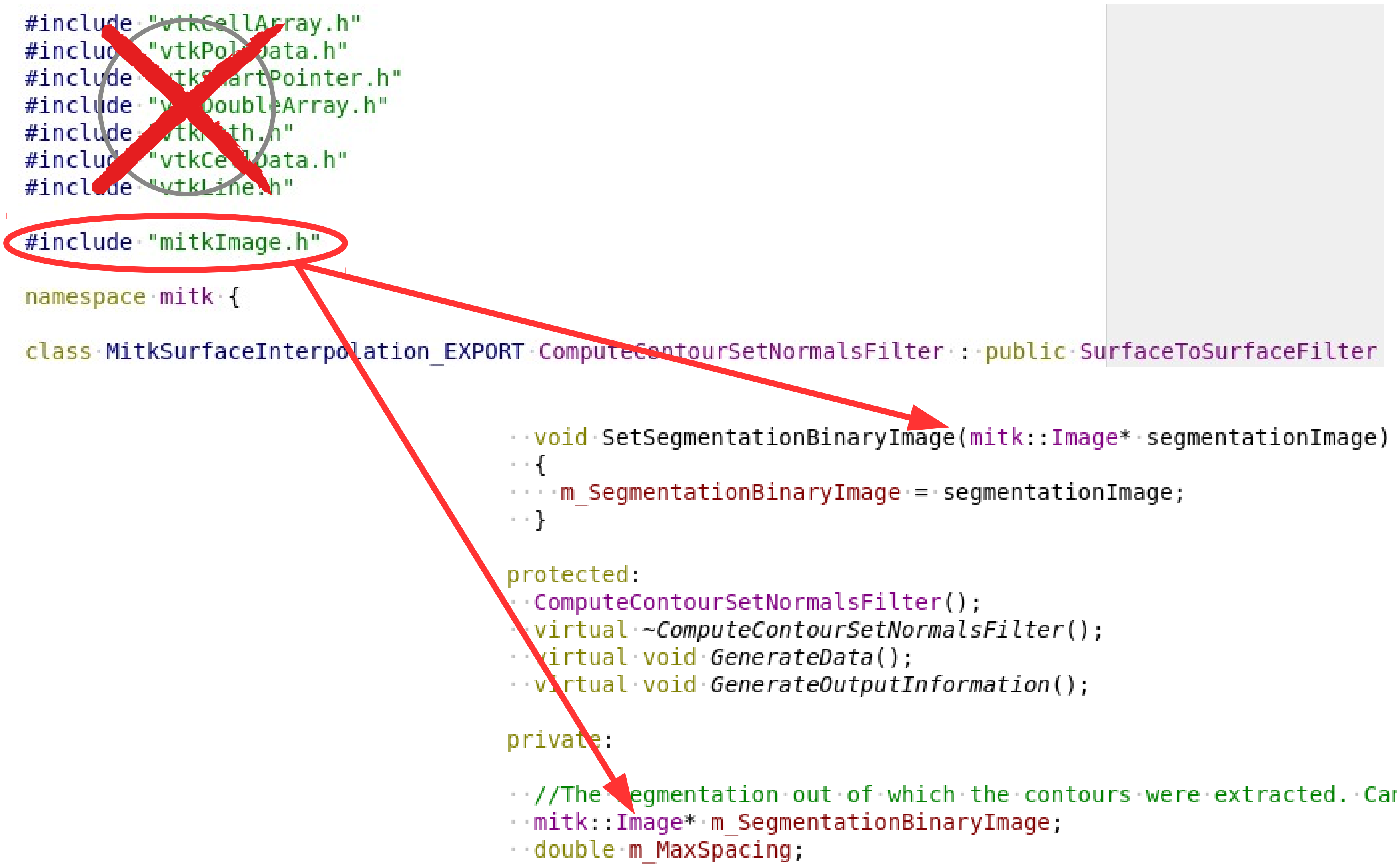
Why should I use it?

- Forward declaration of a class:
  - Can replace an #include
    - Reduces compile time: forward declarated class isn't included so no „chain reaction“ of files you don't need
    - Best practise:

      #include in .cpp file
  - Solves circle dependencies

    ... sometimes

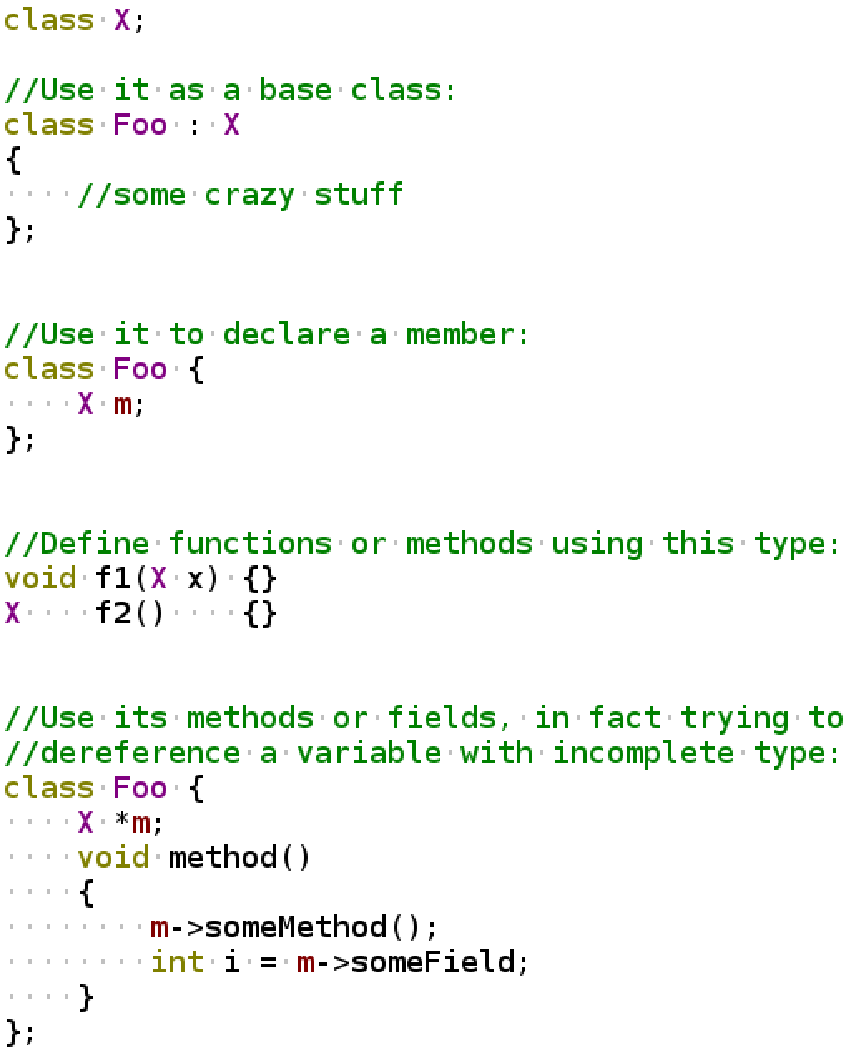
And in MITK?



```
#include "vtkCellArray.h"
#include "vtkPolyData.h"
#include "vtkSmartPointer.h"
#include "vtkDoubleArray.h"
#include "vtkMath.h"
#include "vtkCellData.h"
#include "vtkLine.h"

#include "mitkImage.h"

namespace mitk {

class MitkSurfaceInterpolation_EXPORT ComputeContourSetNormalsFilter : public SurfaceToSurfaceFilter
```

```
  void SetSegmentationBinaryImage(mitk::Image* segmentationImage)
  {
    m_SegmentationBinaryImage = segmentationImage;
  }

protected:
  ComputeContourSetNormalsFilter();
  virtual ~ComputeContourSetNormalsFilter();
  virtual void GenerateData();
  virtual void GenerateOutputInformation();

private:

  //The segmentation out of which the contours were extracted. Car
  mitk::Image* m_SegmentationBinaryImage;
  double m_MaxSpacing;
```

But you cant use it if ...

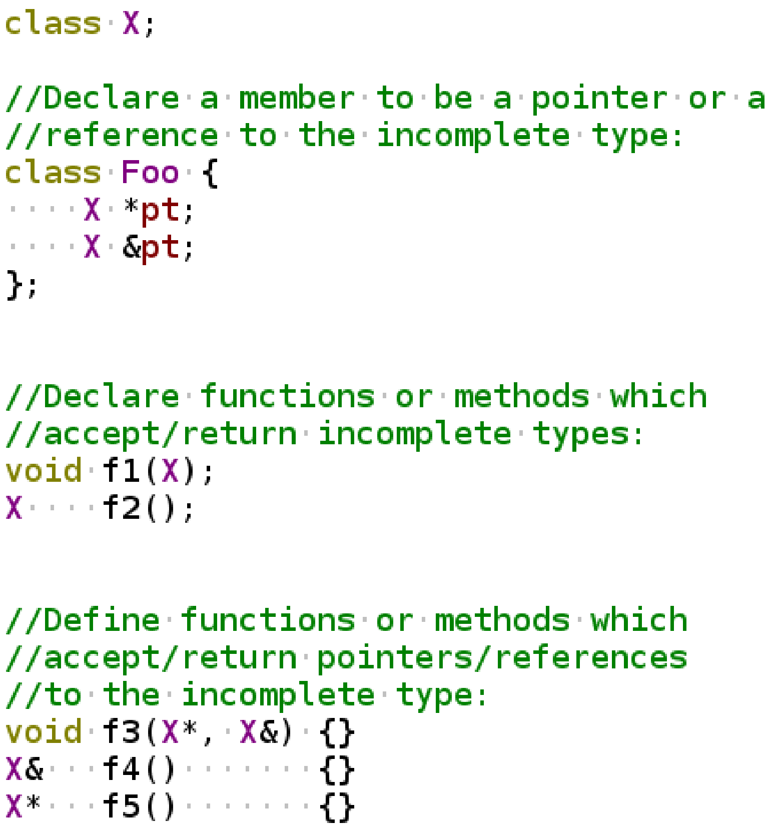


```
class X;

//Use it as a base class:
class Foo : X
{
    //some crazy stuff
};


//Use it to declare a member:
class Foo {
    X m;
};


//Define functions or methods using this type:
void f1(X x) {}
X    f2()    {}


//Use its methods or fields, in fact trying to
//dereference a variable with incomplete type:
class Foo {
    X *m;
    void method()
    {
        m->someMethod();
        int i = m->someField;
    }
};
```

Reason:

The compiler doesn't know the structure of class X

That are good cases:

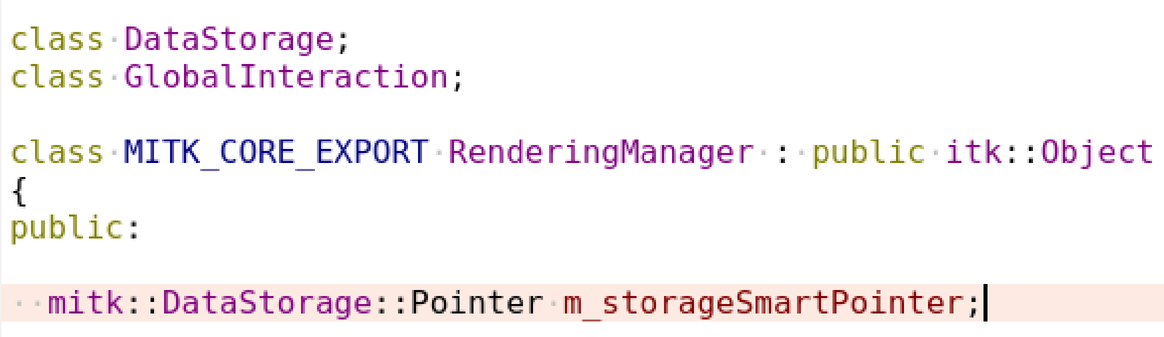


```
class X;

//Declare a member to be a pointer or a
//reference to the incomplete type:
class Foo {
    X *pt;
    X &pt;
};


//Declare functions or methods which
//accept/return incomplete types:
void f1(X);
X    f2();


//Define functions or methods which
//accept/return pointers/references
//to the incomplete type:
void f3(X*, X&) {}
X&   f4()       {}
X*   f5()       {}
```

And why is this possible?!

A pointer always needs the same memory
No information about the object is needed

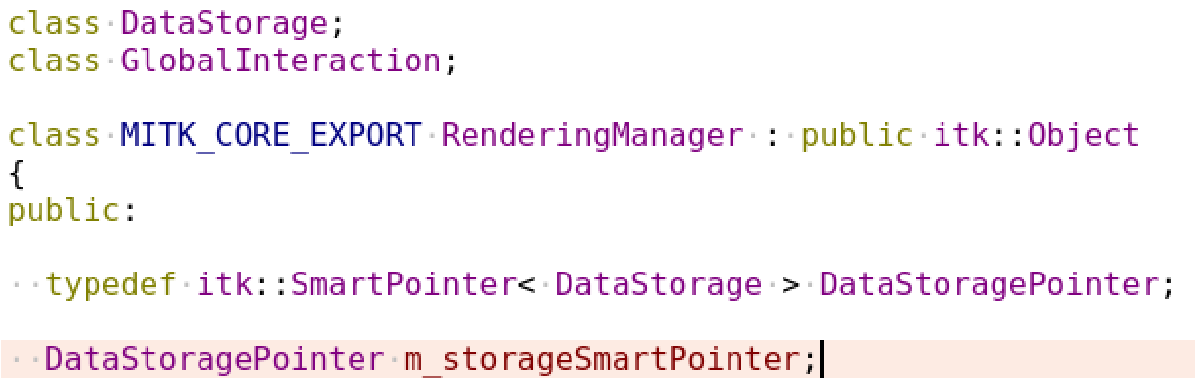
Forward declaration and SmartPointers



```
class DataStorage;
class GlobalInteraction;

class MITK_CORE_EXPORT RenderingManager : public itk::Object
{
public:

  mitk::DataStorage::Pointer m_storageSmartPointer;
```

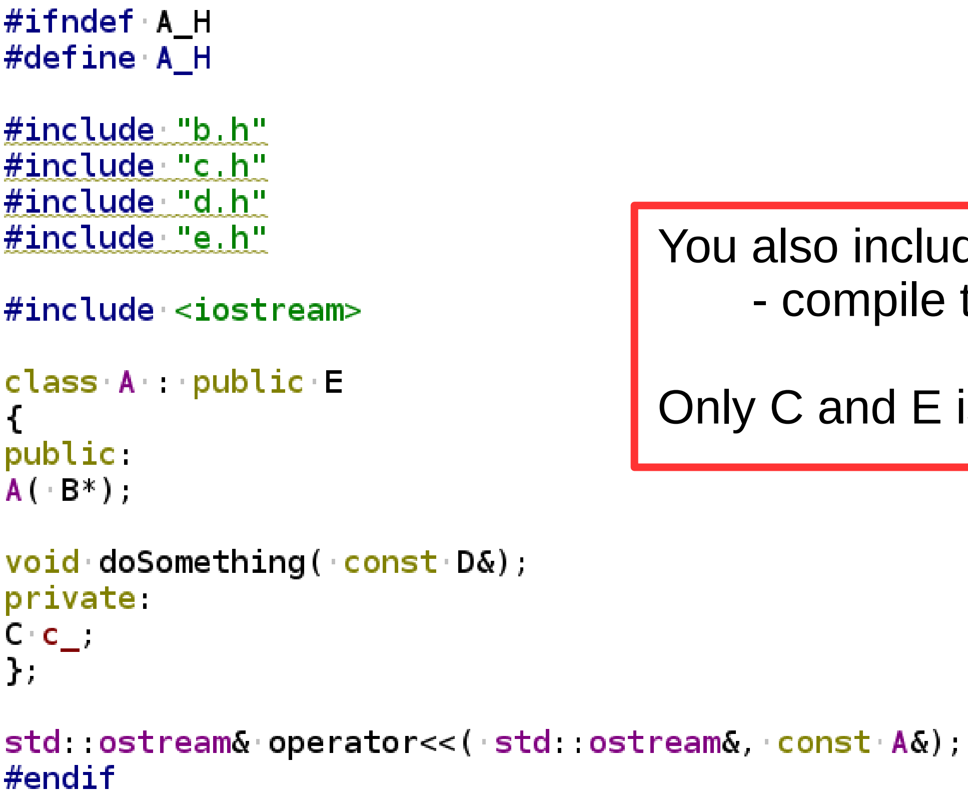
✗

✓



```
class DataStorage;
class GlobalInteraction;

class MITK_CORE_EXPORT RenderingManager : public itk::Object
{
public:

  typedef itk::SmartPointer< DataStorage > DataStoragePointer;

  DataStoragePointer m_storageSmartPointer;
```

Example ... a bad one



```
#ifndef A_H
#define A_H

#include "b.h"
#include "c.h"
#include "d.h"
#include "e.h"

#include <iostream>

class A : public E
{
public:
A( B*);

void doSomething( const D&);
private:
C c_;
};

std::ostream& operator<<( std::ostream&, const A&);
#endif
```

You also include b, c, d, e and iostream
- compile time

Only C and E is rly needed

Conclusion

If you can use it,
do it!

Vielen Dank!

DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

50 Jahre – Forschen für
ein Leben ohne Krebs