C++11 in MITK

auto

Changes in Build System

- Last week presented by Sascha

- Miminal set of C++11 features enabled in MITK
  - auto
  - nullptr
  - override

auto

- Deduce type of variable from its initializer expression

- Provides [1]
  - Eeasier coding / maintanability
  - Correctness
  - Performance (no implicit conversion)
  - Usability
  - Convenience

[1] http://programmers.stackexchange.com/questions/180216/does-auto-make-c-code-harder-to-understand

Easier Coding

- Correct type deduced (automatically)

```
void foo(const std::map<int, std::string>& x)
{
        for ( auto it = x.begin() ; it != x.end() ;
it++ ) { //.... }
}
```

- What type is `it` ?

```
std::map<int, std::string>::const_iterator
```

Easier Coding

- When using new

```
SomeType<OtherType>::SomeOtherType pObject =
                                        new SomeType<OtherType>::SomeOtherType();

auto pObject = new SomeType<OtherType>::SomeOtherType();
```

- Complex type deduction becomes simple

```
template<class T, class U>
void multiply(const vector<T>& vt, const vector<U>& vu)
{
  // ...
  auto tmp = vt[i]*vu[i];
  // ...
}
```

  - The compiler knows the type automatically with T and U

Prefer **auto** to Explicit Type Declarations

- We use it „implicitely“

```
int f = foo();
size_t b = bar();
size_t total = f + b;

std::cout << total << "\n";   => std::cout << (foo() + bar())<< "\n";
```

- Platform-specific size_type

```
std::vector<int> v
unsigned int sz = v.size() // should be std::vector<int>::size_type

auto sz = v.size()
```

http://adamromanek.blogspot.de/2014/11/the-auto-keyword-versus-portability.html

Some **rather don‘ts**

- Use **auto** only if the code remains readable
- Explicit types can be useful

```
const size_t max_size = 100;
for ( auto x = max_size; x > 0; --x )  // unclear. could lead to the errors
{}                                     // since max_size is unsigned
```

- when using not very well-known types

```
//bad : auto decreases readability here
auto obj = ProcessData(someVariables)
```

Summary

*„Use **auto** by default, unless you want have a type conversion to an explicit type“*

[Herb Sutter]

*„Explicit type specification is a form of redundancy that can be removed by **auto**, however it also provides a form of* double-checking *for your code“*

[Andrei Alexandrescu]

http://channel9.msdn.com/Shows/Going+Deep/C-and-Beyond-2012-Scott-Andrei-and-Herb-Ask-Us-Anything