Cp3--
Modular programming support for C++

Baltasar García Perez-Schofield

SI1
Faculty of Computer Science
University of Vigo

http://webs.uvigo.es/jbgarcia/prjs/

C++ for lecturing programming

- The trend of using programming languages designed for teaching, such as Pascal, has vanished.
- Nowadays, it is expected to lecture programming by means of a programming language used in the industry.
- Of course, this raises some problems. Many of these languages are not ideal for lecturing. Some others hide too much the internals of programming.

Approaches for lecturing programming

- Firstly the imperative paradigm, then the object-oriented paradigm.
  - Typically, first C, then C++
- Objects first
  - Typically Java. Sometimes the course is introduced by a learning environment such as BlueJ.
- Pure approaches: Lisp or Scheme.

Modular programming

- When should modular programming be taught?
  - For some programming languages, like Java, you have to mandatorily create one file for each class, many environments automatize this.
  - For many others, specially C-like, modular programming is an obscure, hand-crafted art.
- Students are either unaware of using modular programming, or haven't been taught to use it, due to its complexity on some programming languages.

Modular programming in C++ for computer science students

- Many programming language characterstics constantly *get in the way.*
- Many faculties just use a single .cpp file for all exercises.

```
// math_module.h
#ifndef MATH_MODULE
#define MATH_MODULE
const double PI = 3.14;
double sqr(double x);
#endif


// math_module.cpp
#include "math_module.h"
double sqr(double x)
   { return x * x; }
```

Modular programming in C++ for seasoned programmers

- Seasoned programmers repeat the same process again and again, though it is error-prone and does not improve productivity.
- The only reason this is done this way is due to the old roots of the C++ programming language. Linkers that time were not specially sophisticated.
- New languages such as Java, C#, or even D (a successor of C++) does not ask you to divide your code in interface and implementation.

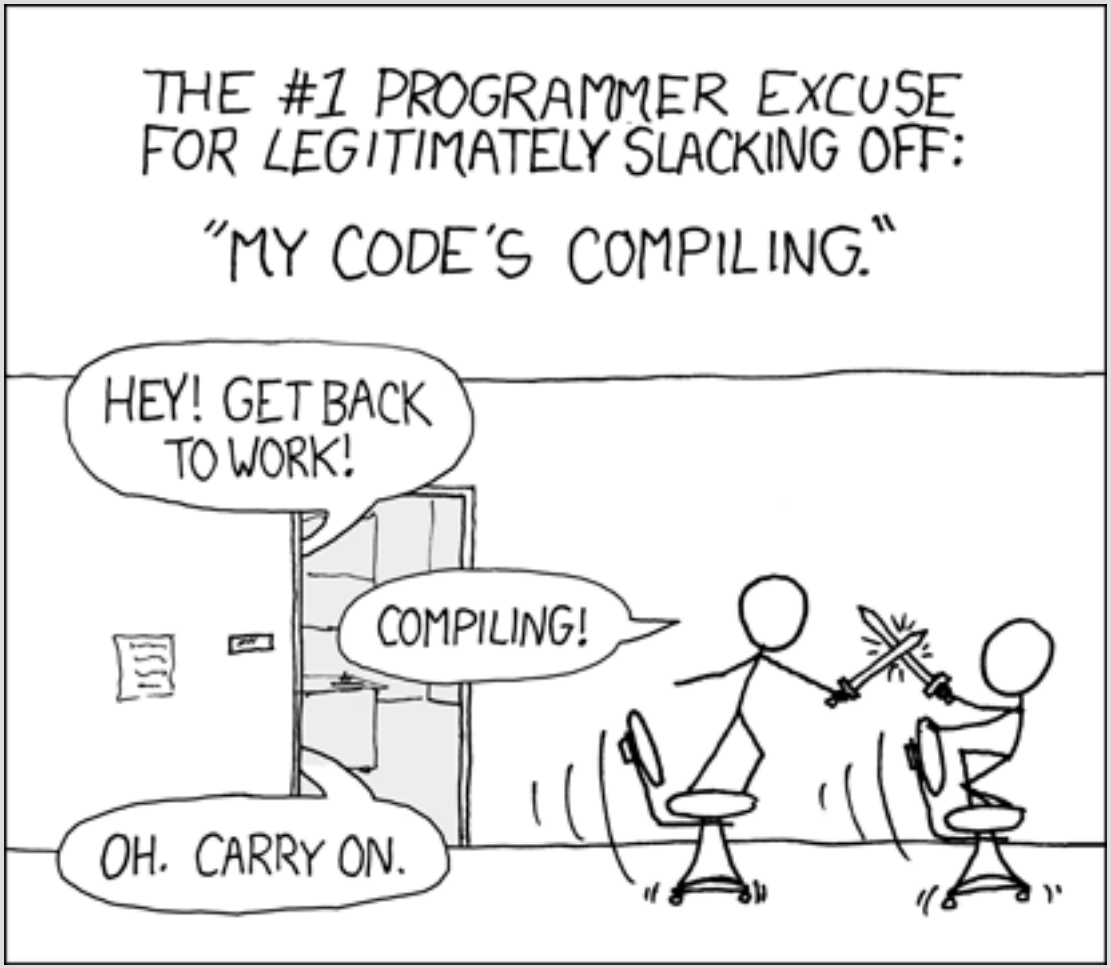
Is it possible to solve this?

- The standard comitee has in modular programming one of its interests for new C++.
- However, it will not make it in the so called C++0x.
- The proposal is a radical change in syntax and use, *a la* Modula - 2.

Is it still possible to solve this?

- It is possible to precede compilation from another program, just a preprocessor, translator or compiler that does the job.
- It will be perceived as a preprocessor for the language.
- There have been many attempts, like **Preprocess**, though they intentionally change the programming language, or are complex to use.

Are we using modern languages?



Cp^3--

- It is a compiler that sits before the compilation of a C++ program.
- It is able to divide the code in the appropriate files, create macro guards, etc.
- It is even able to be more or less strict in aspects of what is allowed on each module, which is useful for teaching.
- It can be either added to the compiler toolchain, or used a code generator.
- Existing files do not need to be modified.

Cp³--

- Code snippets like the previous ones are extremely simplified.

```
// math.mpp
namespace Math {


const double PI = 3.14;


double sqr(double x)
{
   return x * x;
}
}
```

Cp^3--

- Code for classes is also simplified.
- Modifiers are now more homogeneous.

```
// point.mpp
class Point {
   inline
   Point(double a, double b)
                : x(a), y( b) {}
   double getX()
      { return x; }
private:
   double x;
   double y;
};
```

Cp^3--

```
// point.h
class Point {
Point(double a,
double b)
: x(a), y( b) {}
   double getX();
private:
   double x;
   double y;
};
```

```
// point.cpp
#include "point.h"

double Point::getX()
{ return x; }
```

Cp^3--

- The strictness level can be chosen from command line switches.
- For example, in level 3 no globals are allowed.

```
// point.mpp
class Point {
   //...
   inline double getX()
      { return x; }
private:
   double x;
   double y;
};
Point p1; // ERROR
```

Conclusions

- It is possible to simplify modular programming, make it become more declarative. Syntax does not need to be changed at all.
- Kowledge about modular programming is desirable for students.
- Simplification of modular programming is desirable for programmers.