INM359: Object-Oriented Programming in C++

Lecturer: Christos Kloukinas*

Welcome to the INM359 module. The corresponding module description can be found here: INM359 Module Description [1]

Office: Room A306B in the College building (same floor as the Programmes Office)

Classes: **Lecture:** Tuesday, 18–20:00 in C345

Labs: Tuesday, 20–21:00 in EG03

Aims

Assuming a basic understanding of object-oriented programming techniques and the Java language, this module will explore object-oriented programming techniques in greater depth.

The module will focus on practical applications, using the C++ language, which is widely used in industry.

Learning Objectives

On successful completion of this course, students should be able to:

- read and modify substantial well-written C++ programs.
- create classes and small programs in C++ that are correct, robust and capable of being understood, reused and modified by others.
- make use of various object-oriented features, including inheritance, multiple inheritance and genericity, to enhance the above qualities.

Course Outline - Sessions

1. Introduction to C++ for Java programmers. Parameter passing by value and reference **[Savitch 4; Stroustrup 2.1–3 (except 2.3.3), 3.2–6 (except 3.5.1), 3.7.1, 5.5; Horstmann 5.7–8]**.
2. Classes in C++ **[Savitch 1, 6.2, 7.1; Stroustrup 2.5.3–4, 2.6, 10.1–6; Horstmann 8]**.
   (Also, **[Stroustrup 24.3.7.2]** for how to turn assertions off)
3. Operator overloading **[Savitch 8.1; Stroustrup 7.4,11; Horstmann 13.4]**.
   More on I/O in C++ **[Savitch 8.3; Stroustrup 21; Horstmann 10.1]**.
4. Genericity (or parametric polymorphism). C++ templates **[Savitch 16.1–2; Stroustrup 13.2–3; Horstmann 13.5]**.
   Introducing the standard template library (STL) **[Savitch 19.1; Stroustrup 16.2.3, 16.3; Horstmann 13.5]**.
5. Pointers and arrays **[Savitch 10.1; Stroustrup 5.1–3; Horstmann 9.7]**.
   Iterators in the STL **[Savitch 17.3, 19.2; Stroustrup 19.1–2]**.
6. Inheritance and dynamic binding in C++ **[Savitch 14, 15; Stroustrup 12; Horstmann 14]**.
   Genericity, pointers and inheritance **[Savitch 16.3; Stroustrup 12]**.
   (For more on templates and subtyping: **[Stroustrup 13.6.3]**)

*http://www.soi.city.ac.uk/~kloukin/

[1] http://vega.soi.city.ac.uk/modcat/mod.cgi?module_id=7145

7. Multiple inheritance **[Stroustrup 15.2]**. **Only Stroustrup for this; the others have nothing.**
8. Memory management: static, stack, dynamic. Construction and destruction of objects **[Savitch 10.3; Stroustrup 10.4; Horstmann 13.2]**.
9. Memory management, continued. Implementing a container class. Program structure, separate compilation, header files **[Savitch 11.1; Stroustrup 9]**.
   (Also **[Stroustrup 13.7]** for compiling templates)
10. Resource management and exceptions **[Savitch 18; Stroustrup 14]**.

Material Provided

Copies of lecture slides and weekly lab exercises (in addition to assignments).

The slides, lab-work and past exams have been kindly provided by Dr Ross Paterson.

Course Text

Walter Savitch, Absolute C++, Addison-Wesley Longman, Reading, Mass, 2002.
`http://www.cse.ucsd.edu/users/savitch/`

Background Reading

- Bjarne Stroustrup, The C++ Programming Language, 3rd edition, Addison-Wesley Longman, Reading, Mass, 1997.
  `http://www.research.att.com/info/bs`
- Cay Horstmann, Computing Concepts with C++ Essentials,2nd edition, Wiley, 1999.
  `http://www.horstmann.com/`
- Bertrand Meyer, Object-Oriented Software Construction, 2nd edition, Prentice Hall, 1997.
  `http://www.inf.ethz.ch/personal/meyer/`

Various

When learning C++, it's helpful to remember that it's 4 languages in one:

- A better C, i.e., a procedural language but more type-safe than C. That's why, for example, the Linux kernel is compiled with a C++ compiler, although it's plain C code; C++'s stronger type-safety helps catch more bugs at compile time.
- A language for data abstraction
- A language for object-orientation
- A language for generic programming (using templates, as in the STL library).

It's also helpful to remember that, unlike Java, C++ was meant to be as close as possible to C and as fast as possible. The latter means that one has access to low-level mechanisms and that the language provides more than one ways of doing things. Programmers choose among the alternatives based on how abstract (or how fast) they want their code to be.

So, C++ can be used both for writing programs which follow all the principles of well structured object-oriented / generic-programming / data-abstraction programs and for writing programs which break almost all of the rules to get maximum speed.

C++'s complexity stems from these two points: its 4-in-1 nature and the fact that it gives you absolute control of the machine.

Concerning the books mentioned in the "Learning resources", please note that the most "authorative" one is Bjarne Stroustrup's book, since he's the author of the C++ language.

For example, if you look at Savitch's "Absolute C++" book, "Further reading" section, he too says that if you should buy a single book on C++, Stroustrup should be your choice. Savitch also refers the reader to other, more advanced, books at certain occasions.

The catch: you may find Stroustrup a bit more difficult to understand, depending on your programming experience (of course you may find the others a bit too easy/boring... ;-)

So, better go to the library and check them out - compare a chapter or two to see with which one you feel more comfortable with.

Some On-Line Books/Documents/References

- **Bruce Eckel's book "Thinking in C++" is available on-line!**
  Original site of the book - HTML format [2]
  Thinking in C++ from PlanetPDF - PDF format [3] **(two volumes - might be easier to read/search)**
- There is a free document called the C++ Annotations [4] - meant to be an introduction to C++ for those who know C. (Original page [5])
- Another free book is Designing Components with the C++ STL [6] ; it focuses in STL, so you might want to look at it later.
- SGI's reference documentation for its version of the STL [7] (Standard Template Library) is also available on-line.
- The author of C++ is Bjarne Stroustrup [8]. You may find his C++ page [9] interesting to visit.

  Among others he has a technical FAQ [10] (Frequently Asked Questions) where you can find answers to such questions as Why are member functions not virtual by default? [11]

  At his page for the 3rd edition of his The C++ Programming Language book [12] he also has a Tour of C++ [13], a Tour of the Standard Library [14] and a an on-line appendix discussing Exception Safety [15].

  At his publications page you can find articles such as:

  - An Overview of the C++ Programming language [16]
  - Learning Standard C++ as a New Language [17]
  - Why C++ isn't just an Object-Oriented Programming Language [18]
  - Abstraction, libraries, and efficiency in C++ [19]
  - C++ Programming Styles and Libraries [20]
  - Programming with Exceptions [21]
  - Sixteen Ways to Stack a Cat [22]. An interesting (and somewhat funny) paper, showing different ways one can program a stack in C++, depending on the constraints (efficiency, abstraction, generality, etc) and the different paradigms (procedural versus object-oriented, etc).

[2] http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html
[3] http://www.planetpdf.com/developer/article.asp?ContentID=6634
[4] http://www.soi.city.ac.uk/~kloukin/cpp/cplusplus.pdf
[5] http://www.icce.rug.nl/documents/cpp.shtml
[6] http://www.informatik.hs-bremen.de/~brey/stlbe.html
[7] http://www.sgi.com/tech/stl/
[8] http://www.research.att.com/~bs/homepage.html
[9] http://www.research.att.com/~bs/C++.html
[10] http://www.research.att.com/~bs/bs_faq2.html
[11] http://www.research.att.com/~bs/bs_faq2.html#virtual
[12] http://www.research.att.com/~bs/3rd.html
[13] http://www.research.att.com/~bs/3rd_tour.pdf
[14] http://www.research.att.com/~bs/3rd_tour2.pdf
[15] http://www.research.att.com/~bs/3rd_safe0.html
[16] http://www.research.att.com/~bs/crc.pdf
[17] http://www.research.att.com/~bs/new_learning.pdf
[18] http://www.research.att.com/~bs/oopsla.pdf
[19] http://www.research.att.com/~bs/abstraction.pdf
[20] http://www.research.att.com/~bs/style_and_libraries.pdf
[21] http://www.research.att.com/~bs/eh_brief.pdf
[22] http://www.research.att.com/~bs/stack_cat.pdf

- The C++ Virtual Library [23]

Becoming a Proficient C++ Programmer

In order to become a proficient C++ programmer you'll have to study other people's code, so that you can learn from their faults and their experience, in just the same way that an architect studies existing buildings. Here's a small list of libraries selected from Stroustrup's page, you can study or use in your programs.

Note that this is provided for information only - it would probably be too hard for you to study any of this right now, better leave this for after the end of the course...

- A list of available C++ libraries known as the C++ libraries FAQ [24].
- Boost.org [25]: A repository for libraries meant to work well with the C++ standard library.
- Doug Schmidt's site with information about a lot of things including the ACE framework [26] and the TAO real-time ORB [27].
  (ACE and TAO are used for programming distributed systems.)
- (Ex-) AT&T Cambridge Laboratory's robust, reliable, efficient, open-source, CORBA2.1 compliant OmniORB [28]
- High-performance numerical libraries provide excellent tests for interesting new programming techniques: The Object-Oriented Numerics Page [29] is a list of libraries, projects, and mailing lists. For example: Blitz++ [30] from U. of Waterloo, MTL [31] from U. of Notre Dame, and ROOT [32] from CERN.

Tools you need

- `g++`
- `cpp` (part of `g++`)
- `ld` (part of `g++`)
- `nm`
- `c++filt`
- `make`
- `gdb`
- `cvs`
- `svn`

More about these can be obtained by using the manual pages and the info pages - here's how you learn how to use these:

- `man man`
- `info info`

If you're using MS Windows, try `cygwin.com` for a set of Unix tools (better install the full thing, to make sure you did not miss something).

[23] `http://www.desy.de/user/projects/C++.html`
[24] `http://www.trumphurst.com/cpplibs1.html`
[25] `http://www.boost.org/`
[26] `http://siesta.cs.wustl.edu/~schmidt/ACE.html`
[27] `http://siesta.cs.wustl.edu/~schmidt/TAO.html`
[28] `http://omniorb.sourceforge.net/`
[29] `http://www.oonumerics.org/oon`
[30] `http://oonumerics.org/blitz/`
[31] `http://lsc.nd.edu/research/mtl`
[32] `http://root.cern.ch/root`