Applications Computational Finance, I

Goals

- Apply C++ to writing code and applications in computational finance
- Design and implement code so that it is accurate and efficient
- Learning how to analyse/design problems and produce running code
- Become a good programmer (can be used in many domains)
- ‘get it working, then get it right, then get it optimised’

2

Review of C++

- Need to check which C++ syntax and knowledge is optimal when developing applications
- Ability to use VS environment and resolve compiler/linker errors ASAP
- Mathematical/numerical/algorithmic/QF knowledge
- The 20-80 rule!

3

C++ Top 10 (1/2) Skills

- How data is created, allocated, used and destroyed
- Different kinds of functions; global, members, function pointers, function objects
- How data and functions combine; namespaces, structs, classes
- Pointers and their uses
- The ‘const’ stuff

4

C++ Top 10 (2/2) Skills

- Creating basic robust classes
- STL; vector, iterators, algorithms; a bit of Boost is also useful
- Design: inheritance, composition
- Single Responsibility Principle (SRP)
- Partition QF applications into loosely coupled subsystems

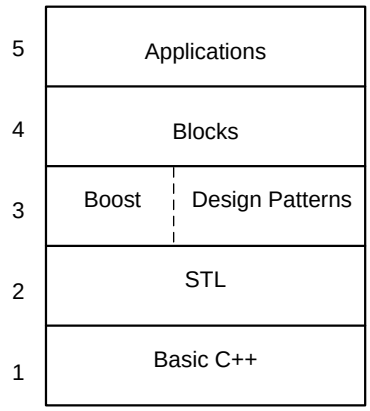
5

Problem Dimensions

- 1. Range of applications in computational finance
- 2. Numerical algorithms
- 3. Software design and implementation
- 4. Using C++ libraries
- 5. C++ and its use in applications
- 6. Debugging and testing

6

Checklist



7

1. Range of Applications

- Main interest is (1-factor) option pricing (equity, fixed income)
- PDE, Monte Carlo, lattice, exact models
- Related applications; calibration, interpolation, numerical algorithms
- Each ‘model’ has its advantages and disadvantages (efficiency, applicability, learning curve)

8

2. Numerical Algorithms and Libraries

- We need various kinds of ‘building blocks’ for application ‘infrastructure’
- Choices are: make yourself or use (from others)
- Resources: Boost, STL, others on the Web
- Related issues: appropriate data structures and numerical linear algebra

9

3. Software Design and Implementation

- Approach application development in phases
- ‘Get it working’ (accurate algorithms and pricers)
- ‘Get it right’ (build in extra functionality)
- ‘Get it optimised’ (OO frameworks, design patterns)
- 10 year/lifetime plan (‘where do you wanna be in 5 years?’)

10

4. Using C++ Libraries

- Outsource as much as possible; building block approach
- Concentrate on ‘core business’
- Generic libraries (e.g. STL, Boost) and more specific libraries (gsl, Boost)
- Not many C++ libraries; plan B is to wrap Fortran code in C and then call from C++

11

5. C++ and its Use in Applications

- C++ is an all-round systems language
- It is used for application development
- Use the most appropriate tools for the job at hand
- ‘Create the Picasso painting, then the frame’ J

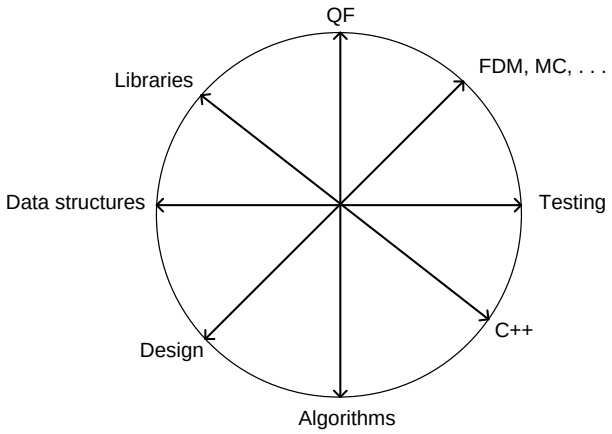
12

6. Debugging and Testing

- Big topic!
- Use appropriate C++ code and take it step-by-step
- Solve compiler and linker errors ASAP
- VS has a debugger (can trace variables etc.)

13

Software Layers



14

Resources

- Books (Clewlow/Strickland, Glasserman, Hull, Wilmott)
- C++/FDM: Duffy
- Web articles on different kinds of pricers
- Build your applications based on published articles

15

Summary

- Overview of what's to come when you start developing C++ applications
- Multi-dimensional/multi-layer problem
- Global preparation..

16