Getting Started with QuantLib

by <u>Vasily Nekrasov</u>

### Preface

Currently there are many books on quantitative finance for graduate students. Most of them are pretty similar and repeat the same stuff on risk-neutral valuation (Black-Scholes model, no-arbitrage, Ito calculus, martingale measure, etc). There are also enough books on numerical finance. These books are more individual, some concentrate on theoretical aspects (convergence, computational efficiency), some focus on programming aspects and on C++ in particular.

This book is different. First of all it is written with practice in mind: **bitter experience clearly tells us that a** *working* **quant not necessarily needs to understand all mathematical nuances<sup>1</sup> but must be able to provide a proper final result and do it** *fast*. The main goal of this book is to teach you this. Another important goal is to give you some hand-on experience with *big* software projects like QuantLib. To my best knowledge, there is no such other book. Additionally, you will also get familiar with version control, doxygen, unit tests and so on. Formally this stuff is not a programming (and thus rarely taught in lectures on programming) but every software engineer needs it in practice<sup>2</sup>.

I deliberately decided to stick to Windows and Visual Studio and not to discuss programming under Linux and Eclipse. These are important issues but the quants usually develop under Windows<sup>3</sup> and porting to Linux (if necessary) is usually assigned to "pure IT" staff. Moreover, Microsoft has recently liberalized the distribution policy and everyone can get Visual Studio Community Edition free of charge<sup>4</sup>. Additionally, students may download VS Professional for free (note that Community and Professional editions seem to be essentially the same). Interestingly, QuantLib was initially Windows-only project and porting to Linux was one of the first community contribution<sup>5</sup>.

Since I am most experienced as fixed income quant, we will mostly concentrated on respective instruments and interest rate models. This is not due to my personal caprice but rather due to the fact that virtually all financial (and many non-financial) institutions have interest rate swap[tion]s in their portfolios. Moreover, starting with even simplest stock models (that imply stochastic processes) may be hard and starting with the calculation of a bond yield shall be easy (however, not as easy as you might believe). Interestingly, the fathers of QuantLib got started at the interest rate desk<sup>6</sup>.

Writing this book I tried to keep the prerequisites as moderate as possible. I assume the basic knowledge of C++ and object oriented programming (the latter is very briefly reviewed). If you do not possess this stuff, you should first read e.g. "C++ from the Ground Up" by Herbert Schildt. On the other and I do not assume any knowledge of design patterns. Those that are relevant for QuantLib, will be discussed in detail. As to stochastic finance, as long as we deal with date and bond arithmetic, you do not need it. In the sense of 80–20 Pareto rule<sup>7</sup> in may be clever for a "pure programmer" to learn only this relatively easy stuff in order to increase the chances to get a job in a bank or corporate treasury. For more advanced topics some

<sup>&</sup>lt;sup>1</sup> Though it never hurts and often is still useful.

<sup>&</sup>lt;sup>2</sup> Probably, the knowledge of this stuff (or lack of it) distinguishes a programmer from a code monkey.

<sup>&</sup>lt;sup>3</sup> Likely because their solutions are often integrated in Excel.

<sup>&</sup>lt;sup>4</sup> http://www.visualstudio.com/en-us/news/vs2013-community-vs.aspx

<sup>&</sup>lt;sup>5</sup> http://www.moneyscience.com/pg/blog/Admin/read/464541/open-source-finance-1-quantlib-an-interview-withluigi-ballabio

<sup>&</sup>lt;sup>6</sup> ibid

<sup>&</sup>lt;sup>7</sup> http://en.wikipedia.org/wiki/Pareto\_principle

knowledge of stochastic finance is required. Ideally, you should first read both volumes of "Stochastic Calculus for Finance" by Steven Shreve and then "Arbitrage Theory in Continuous Time" by Tomas Björk. If you want to quickly get your hands dirty, read "Financial Calculus: An Introduction to Derivative Pricing" by Baxter and Rennie. This book omits many details but provides a sufficient overview for lazybones. And those, who want to dig deeply and dwell on details should have a look at my LIBOR market model tutorial in Appendix B.

Finally, it is worth telling how I got started with programming and with QuantLib. Well, my first experience was with Basic on KOPBET (Russian 8-bit computer) in the age of 12. Then I learnt Turbo Pascal on i286 in school and Z80 assembler on ZX Spectrum. In 1997 my family could to the first time afford a "serious" computer (it was P166 MMX). By that time the Internet came to Russia and I quickly learnt web programming by myself (at first Perl then PHP) and (being a student) I moonlighted as a web developer. In 1999 I got started with Java ... which turned out to be pretty hard. Object oriented programming was something radically new. Additionally, I encountered event driven model, multithreading and so on. The literature was scarce by that time and the most of freely available tutorials were obsolete and considered Java 1.0, whereas I got started with Java 1.1 and then Java 1.2...

As a result, I was able to write simple applets but it took a couple of years before I *really* understood the OOP. I also read a couple of books on C++ but did not really used it. After I graduated from the university in 2002 I worked as web developer till I moved to Germany in 2005 to make my Master of financial engineering. University programs in Germany are usually very theoretic and mine was no exception. So I dwelt upon theory (including fine measure-theoretic aspects), which I do not regret since if you miss it in the university, you will hardly get a chance to catch up. But since I got few computational tasks, the programming was neglected. Only as I started with my master thesis I revived my programming skills because both theoretical and numerical results were required. By that time I heard about QuantLib (version 0.8.0). I attempted several time to build it from sources and finally got it. However, the installation of the boost binaries was only partially successful and the unittest suite did not work. Analogously to my Java case, I encountered too many new stuff with QuantLib: boost libraries, design patterns, date arithmetic... but now I was very short of time and thus postponed it for a while.

After completing my thesis I got a job as quant developer. I recalled and complemented my C++ knowledge. My main duty was the maintenance and enhancement of a calculation engine ("Rechenkern" in German). This engine was primarily developed to calculate loans and mortgages but it also had some functions to price derivatives. So I used QuantLib to check the calculation results from this Rechenkern. I must note that by that time I did not deeply understood QuantLib. I merely looked how it is done in unit tests then often just put my values and got the result. The situation changed as Dimitri Reiswich's excellent tutorials were published online. They gave me a holistic view of QuantLib.

Later I changed to the German Finance Agency and used QuantLib for model risk validation. I also wanted to implement the HPSn model in QuantLib<sup>8</sup>, which was developed by my front office colleagues. However, there was little interest to my initiative, so I just implemented a very alpha version that was really not worth checking-in to QuantLib repository.

Now, as you know my QuantLib experience, you will likely understand why I am not going to teach you how to contribute to QuantLib. This is a purpose of Luigi Ballabio's book<sup>9</sup> and he definitely can achieve it better than me. I am going to teach you to use the already available QuantLib functionality (which is really huge) and how to write [quick and dirty] extensions that solve *your* problems.

<sup>&</sup>lt;sup>8</sup> http://www.yetanotherquant.com/QuantLib/ImplementingHPS/

<sup>&</sup>lt;sup>9</sup> http://implementingquantlib.blogspot.de/p/the-book.html

# Chapter 1: Building QuantLib and setting up developer toolbox

Building QuantLib is not an easy tasks. I know enough quants that gave up QuantLib because they failed to [quickly] compile it from sources. Thus let us discuss the installation process in detail. First of all you need to install Visual Studio. By the time I wrote this book the current version was VS2013. I highly recommend you to install a community edition, which, as I have already said, is free. VS Express will also do but it does not support plug-ins like Visual Assist (see below).

As the next step you need to build boost libraries. These libraries are "one of the most highly regarded and expertly designed C++ library projects in the world."<sup>10</sup> To build them do the following:

- 1. Download the source code from www.boost.org. Current version is 1.57.0. Unpack to C:\
- 2. Start Visual Studio, go to TOOLS -> Visual Studio Command Prompt (Figure 0.1)
- 3. Change directory to C:\boost\_1\_57\_0\tools\build
- 4. Run bootstrap.bat (this will build b2.exe)
- 5. Copy b2.exe to C:\boost\_1\_57\_0 and run it there (Figure 0.2)
- 6. Drink coffee (if you use older version of Visual Studio or boost, you might also have a lunch) Note that boost evolves quickly and the build-tools also do. A couple of years ago bjam.exe was to be used but currently it is replaced with b2.exe. That's why always have a look at http://www.nuclearphynance.com/Show%20Post.aspx?PostIDKey=152032

In this thread I will provide the current setup instructions for boost.

Now we may build QuantLib. Current version is 1.4.1, which supports VS2012 but not VS2013. Fortunately, it is not a problem, just open QuantLib\_vc11.sln in VS2013 and it will automatically convert it<sup>11</sup> (Figure 0.3). Most likely, VS2013 will be officially supported when this book gets published.

As the next step you need to set up the boost path. In Solution Explorer select all projects,

select Properties -> Configuration Properties -> VC++ Directories and add C:\boost\_1\_57\_0 to Include Directories and C:\boost\_1\_57\_0\stage\lib to Library Directories (Figure 0.4). Press F7 to build solution (you may again go drink coffee).

In principle, we can already start with QuantLib but from practical point of view we are not yet done with toolbox setup. First of all I highly recommend you to install the Visual Assist<sup>12</sup>, a Visual Studio plug-in, which greatly enhances its functionality. It allows you to quickly switch between header (.hpp) and .cpp files (how may Microsoft have missed this feature?!), flexibly search for files in your project by keywords (Figure 1.5), find all references to a variable and much more. Visual Assist is not cheap but there are discounts for private users and students. But if you are a really poor student you may alternatively install VSAid<sup>13</sup>. At least it allows a quick switch between .hpp and .cpp, which is really indispensible. Note that neither Visual Assist nor VSAid work with Visual Studio Express.

Further you should install a **version control** system. I am a big fan of Subversion (aka SVN). QuantLib itself used SVN before migrating to Git. However, Git may mostly be advantageous for big "geographically dissipated" teams. For a "one-or-two developers team" SVN should be fine.

<sup>&</sup>lt;sup>10</sup> Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

<sup>&</sup>lt;sup>11</sup> Such straightforward conversion was not always the case. For example, as one wanted to convert QuantLib 1.2.1 from VS2010 to VS2012 a quick and dirty trick was to replace error "unknown Microsoft compiler" with define QL\_LIB\_TOOLSET "vc100" in D:\sandbox\QuantLib-1.2.1\ql\auto\_link.hpp.

<sup>&</sup>lt;sup>12</sup> http://www.wholetomato.com/features/default.asp

<sup>&</sup>lt;sup>13</sup> http://www.brocksoft.co.uk/vsaid.php

| FILE EDIT VIEW DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP |   |
|--|---|
| 🔆 😋 🚽 💿 🛛 🎦 👻 🔛 🗳 👘 🤟 🦿 🤇 Windows Phone 8.1              | • |
| Solution Explorer de Attach to Process Ctrl+Alt+P        |   |
| Connect to Database                                      |   |
| Connect to Server  |   |
| S <sup>+</sup> Add SharePoint Connection                 |   |
| Connect to Microsoft Azure Subscription                  |   |
| SQL Server   | ۲ |
| الله Code Snippets Manager Ctrl+K, Ctrl+B                |   |
| Choose Toolbox Items                                     |   |
| Add-in Manager   |   |
| NuGet Package Manager                                    | • |
| Extensions and Updates                                   |   |
| Create GUID  |   |
| Error Lookup   |   |
| PreEmptive Dotfuscator and Analytics                     |   |
| Spy++  |   |
| Spy++ (x64)  |   |
| ILDasm   |   |
| Visual Studio Command Prompt                             |   |
| WCF Service Configuration Editor                         |   |
| External Tools   |   |
| Import and Export Settings                               |   |
| Customize  |   |
| Options  |   |

Unfortunately, I cannot teach you version control in detail. But there are enough tutorials in Internet. I just briefly tell you that a proper version control allows you to track the changes, compare versions, rollback to previous version if the current version is erroneous and much more. No serious software project can do without version control<sup>14</sup>. Writing this book I also use SVN: initially a version control was applicable to text files only, but now it can be used for MS Word documents as well (Figure 0.6) and even for the graphic data!

The easiest way to install SVN is to download respective stack from bitnami<sup>15</sup>. This is a server side part, which by default will be available under http://localhost/subversion/

<sup>&</sup>lt;sup>14</sup> Well, I know a bank, whose quants were advanced enough to use QuantLib but they had no version control due to highly bureaucratic IT policy by this bank.

<sup>&</sup>lt;sup>15</sup> https://bitnami.com/stack/subversion

Additionally, you need an SVN client, I highly recommend TortoiseSVN<sup>16</sup>. If you are confused, well, you may also work without a version control. But I highly recommend you to learn it, you *will* need it if you are going to program professionally!



<sup>&</sup>lt;sup>16</sup> http://tortoisesvn.net

| Upgrade VC++ Cor  | npiler and Libraries   |   |
|---|--|---|
| The following project(s) uses<br>the Microsoft Visual Studio 2<br>be automatically upgraded t<br>will require the corresponding   | ; an earlier version of the Visual C++ compiler and lib<br>:013 compiler and libraries. Any managed or native co<br>o target .NET Framework 4.5. Note: If you do not upg<br>ig version of Visual Studio to be installed. | oraries. The project(s) will be upgraded to u<br>ode project(s) using C++/CLI extensions w<br>grade the project(s), building your project(s |
| ☑\QuantLib-1.4.1\Quant  | Lib_vc11.vcxproj   |   |
| ✓\test-suite\testsuite_vci  | 11.vcxproj   |   |
| \Swap\Swap_vc11.vcxp  | roj  |   |
| \EquityOption\EquityO   | ption_vc11.vcxproj   |   |
| \ConvertibleBonds\ | nvertibleBonds_vc11.vcxproj  |   |
| \FRA\FRA_vc11.vcxproj   | j  |   |
| \Repo\Repo_vc11.vcxp  | roj  |   |
| Replication\Replicatio  | n_vc11.vcxproj   |   |
| \BermudanSwaption\B   | ermudanSwaption_vc11.vcxproj   |   |
| \DiscreteHedging\Disc   | reteHedging_vc11.vcxproj   |   |
| \FittedBondCurve\Fitte  | dBondCurve_vc11.vcxproj  |   |
| \CallableBonds\Callabl  | eBonds_vc11.vcxproj  |   |
| \CDS\CDS_vc11.vcxpro  | j  |   |
| \Bonds\Bonds_vc11.vc  | xproj  |   |
| \MarketModels\Market  | :Models_vc11.vcxproj   |   |
|   |  | OK Cancel   |

### Figure 0.3 Converting QuantLib from VS2012 to VS2013

| 🌀 - 이 🖥 - 😩 💾 🔐 🔊 - 은 - 🕨 Local Wind  | ows Debugger 👻 Auto 🗣   | v ⊘ v Debug v Win32   |   |
|---|---|---|---|
| Solution Explorer     - ↓ ×       ○ ○ △ │ ○ - ≠ □ │ ▶ -       Search Solution Explorer (Ctrl+ū)       > Solution 'QuantLib_vc11' (L5 projects)       > ③ BermudanSwaption   | Property Pages Configuration: Active(Debug) Common Properties   | Platform:   | Active(Win32) Configuration Manager   |
| Image: Second | References<br>Configuration Properties<br>General<br>Debugging<br>VC++ Directories<br>C/C++<br>XML Document Generator<br>Browse Information<br>Build Events<br>Custom Build Step<br>Code Analysis | Executable Directories<br>Include Directories<br>Reference Directories<br>Library Directories<br>Library WinRT Directories<br>Source Directories<br>Exclude Directories | \$(VC_ExecutablePath_x86);\$(WindowsSDK_ExecutablePath)<br>C:\boost_1_57_0;\$(IncludePath)<br>\$(VC_ReferencesPath_x86);<br>C:\boost_1_57_0\stage\lib;\$(LibraryPath)<br>\$(WindowsSDK_MetadataPath);<br>\$(VC_SourcePath);<br>\$(VC_IncludePath);\$(WindowsSDK_IncludePath);\$(MSBuild |
| ▶  Testsuite  | 2013  | Library Directories<br>Path to use when searching for<br>environment variable LIB.  | library files while building a VC++ project. Corresponds to OK Cancel Apply   |

|   | /mbol                             | Definition  |   |
|---|-----------------------------------|---|---|
| Q | LiborMarketModelProcessTest.testM | void LiborMarketModelProcessTest::testMonteCarloCapletPricing(){} |   |
| 3 | LiborMarketModelProcessTest.testM | typedef LowDiscrepancy::rsg_type rsg_type                         |   |
|   |                                   |   |   |
|   |                                   |   |   |
|   | 3OR Monte                         |   | O |

| Zusammenfassung: 6 Überarbeitungen 🧖 🌸 🗴                    | Veralichenes Dokument  |                      |   | x   | Originaldokument (BookQuantl ib.docx-rev1.svn000.tmn - X  |
|---|--|----------------------|---|---|---|
| Einfürungen 2 Läschurgen - 2 Verschick                      | A Dup beatstran bat (bis will 1.11)  | h2 -                 |   | Ča.   | Dup boottop bat/birull build boots  |
| Einrugungen: 3 Loschvorgange: 2 Verschlebungen: 0           | <ol> <li>Kun bootstrap.bat (this will build</li> <li>Copy b2.exe to C·\boost 1 57 0</li> </ol> | 02.ex                | (e)<br>run it there (Figure 1-2)                      |   | <ol> <li>Kun bootstrap.bat (this will build b2.exe)</li> <li>Copy b2.exe to C:\boost 1, 57, 0 and run it</li> </ol> |
| Formatierung: 1 Kommentare: 0                               | <ol> <li>Drink coffee (if you use older very</li> </ol>  | rsion o              | of Visual Studio or boost, vou might                  |   | there (Figure 1.2)  |
| Änderungen und Kommentare des                               | also have a lunch)   |                      | seest, fearinght                                      |   | <ol> <li>Drink coffee (if you use older version of</li> </ol>   |
| Hauptdokuments  | Note that boost evolves quickly and the k  | build-               | tools also do. A couple of years ago                  |   | Visual Studio or boost, you might also have a   |
| Eingefügt Comparison  | bjam.exe was to be used but currently it   | is rep               | places with b2.exe. That's why                        |   | lunch)  |
| I am a big fan of Subversion (aka SVN).                     | always have a look at http://www.nuclea  | arphyı               | nance.com/Show%20Post.aspx?                           |   | Note that boost evolves quickly and the build-tools   |
| QuantLib itself used SVN before migrating to                | In this thread I will provide the current or   | etuni                | instruction for boost                                 |   | also do. A couple of years ago blam.exe was to be<br>used but currently it is replaces with b2 eve. That's          |
| Git. However, Git may mostly be                             | and an easily an provide the current se  | - tap i              |   |   | why always have a look at   |
| advantageous for big "geographically                        | Now we may build QuantLib. Current ve  | ersion               | is 1.4.1, which supports VS2012 but                   |   | http://www.nuclearphynance.com/Show%  |
| developers team's VN should be fine.                        | not VS2013. Fortunately, it is not a pr  | roblei               | m, just open QuantLib_vc11.sln in                     |   | 20Post.aspx?PostIDKey=152032  |
| Gelöscht  | VS2013 and it will automatically convert it  | t <sup>11</sup> (Fig | gure 1.3). Most likely, VS2013 will be                |   | In this thread I will provide the current setup   |
| Comparison  | orricially supported when this book gets p   | publis               | ined.   |   | instruction for boost.  |
|   | projects.  | JUUST                | pach, in solution explorer select all                 |   | Now we may build QuantLib. Current version is   |
| Formatiert Comparison                                       | select Properties -> Configuration Pro   | operti               | ies -> VC++ Directories and add                       |   | 1.4.1, which supports VS2012 but not VS2013.  |
| Block   | C:\boost_1_57_0  |                      | to  |   | Fortunately, it is not a problem, just open   |
| Fingefijgt Comparison                                       | Include Directories and C:\boost_1_57_0  | 0\sta                | ge\lib to Library Directories (Figure                 |   | QuantLib_vc11.sln in VS2013 and it will automatically   |
| Unfortunately, I cannot teach you version                   | 1.4). Press F7 to build solution (you may again go drink coffee).                              |                      |   |   | convert it** (Figure 1.3). Most likely, VS2013 will be<br>officially supported when this book gots published        |
| control in details. But there are enough                    |  |                      |   |   | As the next step you need to set up the boost path  |
| tutorials in Internet. I just briefly tell you that         | In principle, we can already start with C  | Quant                | Lib but from practical point of view                  |   | In Solution Explorer select all projects,   |
| a proper version control allows you to track                | we are not yet done with toolbox setu  | ip. Fir              | st of all I highly recommend you to                   |   | select Properties -> Configuration Properties ->  |
| the changes, rollback to previous version if                | install the Visual Assist <sup>12</sup> , a Visual Stud  | dio p                | lug-in, which greatly enhances its                    |   | VC++ Directories and add C:\boost_1_57_0 to   |
| more. No serious software project can do                    | tunctionality. It allows you to quickly swit   | itch be              | etween header (.hpp) and .cpp files                   |   | Include Directories and C:\boost_1_57_0\stage\lib   |
| without version control <sup>14</sup> . Writing this book I | project by keywords (Figure 1.5), find all   | eature<br>I refer    | erry, nexibly search for files in your                |   | solution (you may again go drink coffee)  |
| also use SVN.   | Visual Assist is not cheap but there are o   | discou               | unts for private users and students.                  |   | solution (you may again go unink contee).   |
| Gelöscht Comparison   | But if you are a really poor student you m   | nay al               | ternatively install VSAid <sup>13</sup> . At least it |   |   |
|   | allows a quick switch between .hpp and .cpp, which is really indispensible. Note               |                      |   |   | In principle, we can already start with QuantLib but  |
| Änderungen der Kopf- und Eußzeile                           | that neither Visual Assist nor VSAid work  | with                 | Visual Studio Express.                                |   | from practical point of view we are not yet done  |
| (ohne)  |  |                      |   |   | with toolbox setup. First of all I highly recommend   |
| Textfeldänderungen  | Further you should install a version cont  | trol s               | vstem. I am a big fan of Subversion                   |   | in, which greatly enhances its functionality. It allows   |
| (ohne)  | (aka SVN). QuantLib itself used SVN be   | fore                 | migrating to Git. However, Git may                    |   | you to quickly switch between header (.hpp) and   |
| Änderungen an Textfeldern in der Kopf-                      | mostly be advantageous for big "geograp  | phical               | lly dissipated" teams. For a "one-or-                 |   | .cpp files (how may Microsoft have missed this  |
| und Fußzeile  | two developers team"   | SVI                  | N should be fine.                                     |   | feature?!), flexibly search for files in your project by  |
| (onne)  |  |                      |   |   | keywords (Figure 1.5), find all references to a   |
| Fubhotenanderungen  | Start Page - Microsoft Visual Studio   |                      |   |   | but there are discounts for private users and   |
| Eingefügt Comparison  | FILE EDIT VIEW DEBUG TEAM  | ТО                   | OLS TEST ANALYZE WINDOW                               |   | students. But if you are a really poor student you  |
| advanced enough to use QuantLib but they had no             | 0-0 10-0 -0 -0   | 2                    | Windows Phone 8.1                                     |   | may alternatively install VSAid <sup>13</sup> . At least it allows a *  |
| version control due to highly bureaucratic IT policy        |  | ®                    | Attach to Process                                     |   | quick switch between .hpp and .cpp, which is really •   |
| by this bank.   | Solution Explorer  | +9                   | Connect to Database                                   |   | indispensible. Note that neither Visual Assist nor  |
| Endnotenänderungen  | G O 🏠 🎽  | T tan                | Connect to Database                                   |   |   |
| (onne)  |  |                      | Add ChareDairt Course t                               |   | Uberarbeitetes Dokument (BookQuantLib - Comparison) X   |
|   |  | ЦT                   | Add SharePoint Connection                             |   | 4. Run bootstrap.bat (this will build b2.exe)   |
|   |  |                      | Connect to Microsoft Azure Subscription               | •   | 5. Copy b2.exe to C:\boost_1_57_0 and run it  |
|   |  |                      | SQL Server  |   | there (Figure 1.2)  |
|   |  | ÷                    | Code Snippets Manager                                 |   | <ul> <li>b. Drink cottee (if you use older version of<br/>Visual Studie or boost, you might also have a</li> </ul>  |
|   |  |                      | Choose Toolbox Items                                  |   | lunch)  |
|   | Add-in Manager   |                      |   | Note that boost evolves quickly and the build-tools |   |
|   |  |                      | NuGet Package Manager                                 |   | also do. A couple of years ago bjam.exe was to be   |
|   |  | 12                   | Extensions and Updates                                |   | used but currently it is replaces with b2.exe. That's   |
|   |  |                      | Croate GUID   |   | why always have a look at   |
|   |  |                      | Free Lookur   |   | 20Post.aspx?PostIDKev=152032  |
|   |  |                      | Error Lookup  |   | In this thread I will provide the current setup   |
|   |  |                      | PreEmptive Dotfuscator and Analytics                  |   | instruction for boost.  |
|   |  |                      | Spy++   |   |   |
|   |  |                      | Spy++ (x64)   |   | Now we may build QuantLib. Current version is   |
|   |  |                      | ILDasm  |   | 1.4.1, which supports VS2012 but not VS2013.  |
|   |  |                      | Visual Studio Command Prompt                          |   | QuantLib vc11.sln in VS2013 and it will automatically   |
|   |  | œ                    | WCF Service Configuration Editor                      |   | convert it <sup>11</sup> (Figure 1.3). Most likely, VS2013 will be  |
|   |  |                      | External Tools  |   | officially supported when this book gets published.   |
|   |  |                      | Import and Export Settings                            |   | As the next step you need to set up the boost path.   |
|   |  |                      | Customize   |   | In Solution Explorer select all projects,   |
|   |  | ø                    | Options   |   | select Properties -> Configuration Properties ->  |
|   | Figure 1.1 Visual Studio Command Prom  | npt                  |   |   | Include Directories and C:\boost 1 57 0\stage\lib   |
|   | Figure 1.1 visual Studio Command Prompt  |                      |   |   | to Library Directories (Figure 1.4). Press F7 to build  |
|   | Unfortunately, I cannot teach you version control in details. But there are enough             |                      |   |   | solution (you may again go drink coffee).   |
|   | tutorials in Internet. I just briefly tell you   | u that               | a proper version control allows you                   |   |   |
|   | to track the changes, rollback to prev   | vious                | version if the current version is                     |   |   |
|   | control <sup>14</sup> . Writing this book Lake use SVM   | SOTTW                | vare project can do without version                   |   | In principle, we can already start with QuantLib but  |
|   | control <sup>14</sup> . Writing this book I also use SVN.                                      |                      |   |   | with toolbox setup. First of all L highly recommend   |
|   | C:\Windows\system32\cmd ava  |                      | Street and  |   | you to install the Visual Assist <sup>12</sup> , a Visual Studio plug-  |
|   | c:\urogram files (x86)\micuoso   | ft v                 | isual studio 12 Averbinied C                          |   | in, which greatly enhances its functionality. It allows   |
|   | ols/build/   |                      |   |   | you to quickly switch between header (.hpp) and   |
|   | C:\boost_1_57_0\tools\build>bo   | otst                 | rap.bat   |   | .cpp files (how may Microsoft have missed this  |
|   | bootstrapping the build engine   |                      |   |   | reature (1), tiexibly search for files in your project by<br>keywords (Figure 1.5), find all references to a        |
|   | Bootstrapping is done. To buil   | ld, r                | un :  |   | variable and much more. Visual Assist is not chean  |
|   | .∖b2prefix=DIR install   |                      |   |   | but there are discounts for private users and   |
|   |  |                      |   | -   | students. But if you are a really poor student you  |

#### Figure 0.6 SVN and MS Word: comparing two versions of this book

Last but not least we install the Doxygen<sup>17</sup>. "Doxygen is de facto the standard tool for generating documentation from annotated C++ sources ... [it] is very useful to *quickly find your way in large source distributions*. Doxygen can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams; all of them are generated automatically".

As I started my career as a quant developer, I had to maintain a big library like QuantLib. Though I previously had some experience with big projects, all of them were websites with a straightforward business logic, directly visible in a browser. Now it was not the case anymore. Fortunately, the library had a rich suite of unittests (Quantlib also has). So as I applied Doxygen to the source code, it extracted a lot of information. The most important were likely the inheritance and call/caller graphs. Let us, for example, have a look at the implementation of the Black-76 model in QuantLib. Doxygen has generated the following inheritance, call and caller graphs:



From the inheritance graph we see that the class *BlackScholesCalculator* inherits from the class *BlackCalculator*. In particular, it means that if we modify the BlackCalculator, this will likely affect the BlackScholesCalculator. Call/caller graphs for the method *elasticity* tells us where this method is called and

<sup>&</sup>lt;sup>17</sup> http://www.stack.nl/~dimitri/doxygen/

which methods it calls. It may be very useful for debugging and we are aware, which methods can be affected if we change something in *elasticity*.

Doxygen has a built-in graph generator but if you want to have them generated nicely, you need to install the Graphviz<sup>18</sup>. It is also recommended to install LaTeX<sup>19</sup>. Doxygen understands LaTeX commands and they are to be found in QuantLib sources.

You should also tune the Doxygen settings in such a way that it generates full graphs. QuantLib developers, themselves, use Doxygen to generate official QuantLib HTML documentation. For the visibility's sake they limit the number of nodes in graphs. However, such visibility may be deceptive, for example, the inheritance graph for the Vasicek model from the official documentation neither tells us that Vasicek belongs to the realm of calibrated models<sup>20</sup> nor shows the members and functions of the classes. An obvious shortcoming of the complete graphs is that they can get illegible. But a 4K monitor<sup>21</sup> can at least partially mitigate this problem.

Ok, we are (almost) done with technical setup, let us run some QuantLib code. Assign project *Bonds* as StartUp project, set a breakpoint at line 240 in Bonds.cpp and start debugging (F5).

| Open Bonds.cpp, press Ctrl + G to and enter 240 in order to jump to the line 240 |          |
|--|----------|
|  | ☞VS Tip☜ |

The variables representation in debugger window is not very clear. In particular, todaysDate is represented as {*serialNumber\_=39706*} and the representation of all shared pointers *boost::shared\_ptr<T>* (soon we will discuss them in detail) is hardly legible. We lay aside the former for a while but can quickly fix the latter. Just install the Natvis visualizer for boost by Arkady Shapkin from

https://visualstudiogallery.msdn.microsoft.com/61f12e7a-bc62-4b2c-b02e-d66014688c2e run the same code again and compare the representation in debugger (Figure 0.9).

Natvis lets you customize the representation of C++ types in the debugger. It is a relatively new technology, which was introduced in VS2012. Before one overwrote the data representations in autoexp.dat (not supported anymore from VS2012). As a *quant* developer you probably should not dwell upon this technical details too much. But it is obviously worth being aware of this technology and installing ready-to-use visualizers, if available.

☞VS Tip☜

<sup>&</sup>lt;sup>18</sup> http://www.graphviz.org

<sup>&</sup>lt;sup>19</sup> http://miktex.org

<sup>&</sup>lt;sup>20</sup> which means that QuantLib lets us fit the model parameters to the current term structure and a set of traded instruments like swaptions or caps.

<sup>&</sup>lt;sup>21</sup> 4K monitors have 3840 x 2160 pixels.



| <pre>     // Separate     // S</pre>  | // Josephiles     // Jos   | <pre></pre>   | <pre>     // supports     // supports     // supports     // supports     // supports     // supports     // support     // su</pre>   |   |
|---|--|---|--|---|
| P     boost::shareptr:Quote dimate(new SimpleQuote(dimQuote));<br>boost::share_ptr:Quote dimate(new SimpleQuote(dimAute));<br>boost::share_ptr:Quote dimate(new SimpleQuote(dimAute));<br>boost::share_ptr:Quote(dimate);<br>dimate (perDecceccccc(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute);<br>boost::share_ptr:Quote(dimAute)   | O     boost::shared_ptr:Quote> dismate(new SimpleQuote(dimQuote));<br>boost::shared_ptr:Quote> dismate(new SimpleQuote(dimQuote));<br>boost::shared_ptr:Quote> dismate(new SimpleQuote(diqQuote));<br>boost::shared_ptr:Quote> dismate(new SimpleQuote(diqQuote));<br>boost::shared_ptr:Quote> sightet(new SimpleQuote(diqQuote));<br>boost::shared_ptr:Quote);<br>boost::shared_ptr:Quote> sightet(new  | ● boost:::bnred_ptr:Quote> dimBate(new SimpleQuote(dimQuote));<br>boost:::bnred_ptr:Quote> dimBate(new SimpleQuote(deqUote));<br>boost:::bnred_ptr:Quote> dimBate(new SimpleQuote(deqUote));<br>boost:::bnred_ptr:Quote> dimBate(new SimpleQuote(dyQuote));<br>boost:::bnred_ptr:Quote> dimBate(new SimpleQuote(dyQuote));<br>boost:::bnred_ptr:Quote> s2yRate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s2yRate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s2yRate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s2yRate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s2yRate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s2yRate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s1%Rate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s1%Rate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s1%Rate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s1%Rate(new SimpleQuote(SyQuote));<br>boost:::bnred_ptr:Quote> s1%Rate(new SimpleQuote(SyQuote));<br>boost::bnred_ptr:Quote> s1%Rate(new SimpleQuote(SyQuote> s1%Rate(new SimpleQuote(SyQuote));<br>boost::bnred_ptr:Quote> s1%Rate(new SimpleQuote(SyQuote> s1%Rate(new SimpleQuote(SyQuote> s1%Rate(new SimpleQuote(SyQuote> s1%Rate(ne   | boost::shared_ptrQuote> dsmBate(new SimpleQuote(dsmQuote));     boost::shared_ptrQuote> dsmBate(new SimpleQuote(dsQuote));     boost::shared_ptrQuote> dsmBate(new SimpleQuote(dsQuote));     boost::shared_ptrQuote> dsmBate(new SimpleQuote(dsQuote));     boost::shared_ptrQuote> dsmBate(new SimpleQuote(s2QUote));     boost::shared_ptrQuote> s3yRate(new SimpleQuote(s3QUote));     boost::shared_ptrQuote> s3yRate(new SimpleQuote(s1SyQuote));     boost::shared_ptrQuote> s1SyRate(new SimpleQuote(s1SyQuote));     double     '''''''''''''''''''''''''''''''''   |   |
| boot:::shared_ptrQuotes dsmate(new SimpleQuote(dsmQuote));<br>boot:::shared_ptrQuotes dsmate(new SimpleQuote(dsmQuote));<br>boot:::shared_ptrQuotes dsmate(new SimpleQuote(dsvQuote));<br>boot:::shared_ptrQuotes s2yMate(new SimpleQuote(dsvQuote));<br>boot:::shared_ptrQuotes s2yMate(new SimpleQuote(s3yQuote));<br>boot:::shared_ptrQuotes s2yMate(new SimpleQuote(s3yQuote));<br>boot:::shared_ptrQuotes s2yMate(new SimpleQuote(s1SyQuote));<br>boot:::shared_ptrQuotes s2yMate(new SimpleQuote(s1SyQuote));<br>boot:::shared_ptrQuotes s1yMate(new SimpleQuote(s1SyQuote));<br>boot:::shared_ptrQuote(new SimpleQuote(s1SyQuote));<br>boot:::shared_ptrQuotes s1yMate(new SimpleQuote(s1S   | boost::shared_ptr:Quote> d3mate(new SimpleQuote(d3mQuote));<br>boost::shared_ptr:Quote> d9mate(new SimpleQuote(d3mQuote));<br>boost::shared_ptr:Quote> 32%Bate(new SimpleQuote(d3mQuote));<br>boost::shared_ptr:Quote> 40%Bate(new SimpleQuote(damQuote));<br>boost::shared_ptr:Quote> 40%Bate(new SimpleQuote);<br>boost::shared_ptr:Quote> 40%Bate(new SimpleQu  | boost:::hnred_ptr:Quute> dsmRate(new SimpleQuute(dsmQute));<br>boost:::hnred_ptr:Quute> dsmRate(new SimpleQuute(dsmQute));<br>boost:::hnred_ptr:Quute> dsmRate(new SimpleQuute(dsmQute));<br>boost:::hnred_ptr:Quute> styRate(new SimpleQuute(dsyQute));<br>boost:::hnred_ptr:Quute> styRate(new SimpleQuute(styQute));<br>boost:::hnred_ptr:Quute> styRate(new SimpleQuute);<br>boost:::hnred_ptr:Quute> styRate(new SimpleQuute);   | boost:::shared_ptrQuote> dsmRate(new SimpleQuote(dsmQuote));<br>boost::shared_ptrQuote> dsmRate(new SimpleQuote(dsmQuote));<br>boost::shared_ptrQuote> dsmRate(new SimpleQuote(dsQuote));<br>boost::shared_ptrQuote> s3yRate(new SimpleQuote(dsQuote));<br>boost::shared_ptrQuote> s3yRate(new SimpleQuote(s3yQuote));<br>boost::shared_ptrQuote> s3yRate(new SimpleQuote(s3yQuote));<br>boost::shared_ptrQuote> s3yRate(new SimpleQuote(s1byQuote));<br>boost::shared_ptrQuote> s3yRate(new SimpleQuote(s1byQuote));<br>boost::shared_ptrQuote> s1byRate(new SimpleQuote(s1b)Quote);<br>boost::shared_ptrQuote> s1byRate(new SimpleQuote);<br>boost::shared_ptrQuote> s1byRate(new SimpleQuote(s1b)Quote);<br>boo   |   |
| boost::shared_ptrQuote> demBate(new SimpleQuote(demQuote));<br>boost::shared_ptrQuote> dsyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> dsyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> ssyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> ssyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> ssyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> ssyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> ssyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote(dsyQuote));<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote(csyQuote));<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>code> Threads Modules Wateh1 (provectione);<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>boost::shared_ptrQuote> sisyMate(new SimpleQuote);<br>code> Threads Modules Wateh1 (provectione);<br>boost::shared_ptrQuote> sisyMate(new Simple   | boost::shared_ptr <quuck> demRate(mew SimpleQuuct(demQute));<br/>boost::shared_ptr<quuck> diyRate(mew SimpleQuuct(diyQuuct));<br/>boost::shared_ptr<quuck> s2yRate(mew SimpleQuuct(diyQuuct));<br/>boost::shared_ptr<quuck> s2yRate(mew SimpleQuuct(diyQuuct));<br/>boost::shared_ptr<quuck> s2yRate(mew SimpleQuuct(diyQuuct));<br/>boost::shared_ptr<quuck> s1SyRate(mew SimpleQuuct(diyQuuct));<br/>boost::shared_ptr<quuck<br>s2yQuuct<br/>double<br/>double<br/>double<br/>double<br/>double<br/>freeAmount<br/>- 92S9081349317831=-001<br/>boost:shared_ptr<quuntlib:<br>b boodfingine<br/>(pr=Doccccccc(ling_count_=??? weak_count; boost:shared_ptr<quantlib:<br>b boost:shared_ptr<quuck double<br="">(pr=Doccccccc(ling_count_=??? weak_count; boost:shared_ptr<quantlib:<br>b boost:shared_ptr<quuck double<br="">(pr=Doccccccc(ling_count_=??? weak_count; boost:shared_ptr<quantlib:<br>b boost:shared_ptr<quuck double<br="">(pr=Docccccccc(ling_count_=??? weak_count; boost:shared_ptr<quantlib:<br>b toodyPuble<br/>(pr=Docccccccc(ling_count_=??? weak_count; boost:shared_ptr<quantlib:<br>b toodyPuble<br/>(pr=Docccccccc(ling_count_=??? weak_count; boost:shared_ptr<quantlib:<br>b toodyPuble<br/>(pr=Docccccccccc) (pr=QuantLib:Dote<br/>(pr=Doccccccc) (pr=QuantLib:Dote<br/>(pr=Doccccccc) (pr=QuantLib:Dote<br/>(pr=Docccccc) (pr=QuantLib</quantlib:<br></quantlib:<br></quantlib:<br></quuck></quantlib:<br></quuck></quantlib:<br></quuck></quantlib:<br></quuntlib:<br></quuck<br></quuck></quuck></quuck></quuck></quuck></quuck></quuck></quuck></quuck></quuck></quuck></quuck></quuck>  | boost::shared_ptr <quote> d@mate(new SimpleQuote(d@quote));<br/>boost::shared_ptr<quote> diymate(new SimpleQuote(diyQuote));<br/>boost::shared_ptr<quote> siymate(new SimpleQuote(diyQuote));<br/>boost::shared_ptr<quote> siymate(new SimpleQuote(syQuote));<br/>boost::shared_ptr<quote> sigmate(new SimpleQuote(syQuote));<br/>boost::shared_ptr<quote> sigmate(new SimpleQuote(syQuote));<br/>boost::shared_ptr<quote> sigmate(new SimpleQuote(syQuote));<br/>boost::shared_ptr<quote> sigmate(new SimpleQuote);<br/>boost::shared_ptr<quote> sigmate(new SimpleQuote);<br/>classes the sigmate(new SimpleQuote);<br/>classes the sigmate(new SimpleQuote);<br/>classes the sigmate(new SimpleQuote);<br/>sigmate(new SimpleQuote);<br/>cl</quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote>  | boost::shared_ptr <quote> dsmRate(new SimpleQuote(dsmQuote));<br/>boost::shared_ptr<quote> dsmRate(new SimpleQuote(dsyQuote));<br/>boost::shared_ptr<quote> s3yRate(new SimpleQuote(s3yQuote));<br/>boost::shared_ptr<quote> s3yRate(new SimpleQuote(s3yQuote));<br/>boost::shared_ptr<quote> s1gyRate(new SimpleQuote(s3yQuote));<br/>boost::shared_ptr<quote> s1gyRate(new SimpleQuote(s3yQuote));<br/>boost::shared_ptr<quote> s1gyRate(new SimpleQuote(s1gyQuote));<br/>boost::shared_ptr<quote> s1gyRate(new Si</quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote>  |   |
| boot:::bhred_ptr <quote> d9mRate (new SimpleQuote(d9mQuote));<br/>boot:::bhred_ptr<quote> s3yRate(new SimpleQuote(d3yQuote));<br/>boot:::bhred_ptr<quote> s3yRate(new SimpleQuote(d3yQuote));<br/>boot:::bhred_ptr<quote> s3yRate(new SimpleQuote(d3yQuote));<br/>boot:::bhred_ptr<quote> s3yRate(new SimpleQuote(d3yQuote));<br/>boot:::bhred_ptr<quote> s1byRate(new SimpleQuote(d3yQuote));<br/>boot::bhred_ptr<quote> s1byRate(new SimpleQuote(dayQuote));<br/>boot::bhred_ptr<quote> s1byRate(new SimpleQuote(dayQuote));<br/>boot::bhred_ptr<quote> s1byRate(new SimpleQuote(dayQuote));<br/>boot::bhred_ptr<quote> s1byRate(new SimpleQuote(dayQuote));<br/>boot::bhred_ptr<quote> s1byRate(new SimpleQuote(dayQuote));<br/>boot::bhred_ptr<quote> s1byRate(new SimpleQuote(dayQuote));<br/>boot::bhred_ptr<simplequote< string="spanator&lt;br">code tring= spanator<br/>code tri</simplequote<></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote>   | boot:::bhred_ptr <quote> d9mRate(new SimpleQuote);<br/>boot:::bhred_ptr<quote> s3yRate(new SimpleQuote);<br/>boot:::bhred_ptr<quote> s3yRate(new SimpleQuote(s3yQuote));<br/>boot:::bhred_ptr<quote> s3yRate(new SimpleQuote(s3yQuote));<br/>boot:::bhred_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));<br/>boot:::bhred_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));<br/>boot::bhred_ptr<quote> s10yRate(new SimpleQu</quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote>  | boost::shared_ptr <quote> diyAste(new SimpleQuote(diyQuote));         // swaps         boost::shared_ptr<quote> siyAste(new SimpleQuote(siyQuote));         boost::shared_ptr       (preducescote);         boost::shared_ptr       (preduote);         boos</quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote>   | boost::shared_ptr <quote> dlyRate(new SimpleQuote(dlyQuote));<br/>boost::shared_ptr<quote> slyRate(new SimpleQuote(slyQuote));<br/>boost::shared_ptr<quote> slyRate(new SimpleQuote(slyRuote));<br/>boost::shared_ptr<quote> slyRate(new SimpleQuote(slyRuote));<br/>boost::shared_ptr<quattlib::quattlib:< th=""></quattlib::quattlib:<></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote>  |   |
| dodstishered_ptr=Quotes       SimpleQuote(s3yQuote));       //was          //was       SimpleQuote(s1yQuote));       //was          //was        //was <t< td=""><td>Doost:::shared_ptr:Quote&gt; s2yRate(new SimpleQuote(s2yQuote));<br/>boost::shared_ptr:Quote&gt; s3yRate(new SimpleQuote(s2yQuote));<br/>boost::shared_ptr:Quote&gt; s1yRate(new SimpleQuote(s15yQuote));<br/>boost::shared_ptr:Quote&gt; s1yRate(new SimpleQuote(s15yQuote));<br/>boost::shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:shared_ptr:QuantLike:<br/>boost:shared_ptr:QuantLike:<br/>boost:<br/>boost:<br/>boost:shared_ptr:QuantLike:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:shared_ptr:QuantLike:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:shared_ptr:QuantLike:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>boost:<br/>b</td><td>doodst::shared_ptr<quote> alykate(new SimpleQuote(s2yQuote));         boost::shared_ptr<quote> s2yRate(new SimpleQuote(s2yQuote));         boost::shared_ptr<quote> s3yRate(new SimpleQuote(s15yQuote));         boost::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));         boost::shared_ptr&lt;</quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></td>       Chronote         class       Call Stack          class           class           class           class           class       <t< td=""><td><i>boost:::shared_ptrQuote&gt; slyRate(new SimpleQuote(slyQuote)); boost::shared_ptrQuote&gt; slyRate(new SimpleQuote(slyQuote)); boodsdyDate= (prQuote) slyRate(new SimpleQuo</i></td></t<></t<> | Doost:::shared_ptr:Quote> s2yRate(new SimpleQuote(s2yQuote));<br>boost::shared_ptr:Quote> s3yRate(new SimpleQuote(s2yQuote));<br>boost::shared_ptr:Quote> s1yRate(new SimpleQuote(s15yQuote));<br>boost::shared_ptr:Quote> s1yRate(new SimpleQuote(s15yQuote));<br>boost::shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:shared_ptr:QuantLike:<br>boost:shared_ptr:QuantLike:<br>boost:<br>boost:<br>boost:shared_ptr:QuantLike:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:shared_ptr:QuantLike:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:shared_ptr:QuantLike:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>boost:<br>b  | doodst::shared_ptr <quote> alykate(new SimpleQuote(s2yQuote));         boost::shared_ptr<quote> s2yRate(new SimpleQuote(s2yQuote));         boost::shared_ptr<quote> s3yRate(new SimpleQuote(s15yQuote));         boost::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));         boost::shared_ptr&lt;</quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote>  | <i>boost:::shared_ptrQuote&gt; slyRate(new SimpleQuote(slyQuote)); boost::shared_ptrQuote&gt; slyRate(new SimpleQuote(slyQuote)); boodsdyDate= (prQuote) slyRate(new SimpleQuo</i>   |   |
| // Swaps  | <pre>// Sample<br/>boost::shared_ptr<quote> s2yRate(new SimpleQuote(s2yQuote));<br/>boost::shared_ptr<quote> s3yRate(new SimpleQuote(s3yQuote));<br/>boost::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));<br/>boost::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));<br/>boost::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));<br/>/***********************************</quote></quote></quote></quote></quote></pre>   | // Samps  | <pre>// Swaps<br/>boost:::shared_ptr<quote> s2yRate(new SimpleQuote(s2yQuote));<br/>boost:::shared_ptr<quote> s3yRate(new SimpleQuote(s3yQuote));<br/>boost:::shared_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));<br/>boost:::shared_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));<br/>boost:::shared_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));<br/>boost:::shared_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));<br/>boost:::shared_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));<br/>boost:::shared_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));<br/>boost::shared_ptr<quote> s10yRate(s10yQuote));<br/>boost::shared_ptr<quote> s10yRate(s10yQuote));<br/>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:d<br>boost::shared_ptr<quattlib:< td=""></quattlib:<></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quattlib:d<br></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></quote></pre>   |   |
| call  | boost:::shared_ptr <quotes< td="">       s3yRate(new SimpleQuote(153yQuote));<br/>boost::shared_ptr<quotes< td="">         boost:::shared_ptr<quotes< td="">       simpleQuote(153yQuote));<br/>boost::shared_ptr<quotes< td="">         boost:::shared_ptr<quotes< td="">       simpleQuote(153yQuote));<br/>boost::shared_ptr<quotes< td="">         boost:::shared_ptr<quotes< td="">       simpleQuote(150yQuote));<br/>boost::shared_ptr<quotes< td="">         boost:::shared_ptr<quotes< td="">       simpleQuote(150yQuote));<br/>boost::shared_ptr<quotes< td="">         costs       ************************************</quotes<></quotes<></quotes<></quotes<></quotes<></quotes<></quotes<></quotes<></quotes<></quotes<>  | Call Stack       Impl_ shared_ptr (Quote) = 33/Rate((new SimpleQuote(25)Quote));<br>boost::shared_ptr (Quote) = 53/Rate((new SimpleQuote(23)Quote));<br>boost::shared_ptr (Quote) > 519/Rate(new SimpleQuote(31)Quote));<br>boost::shared_ptr (Quote) > 519/Rate(new SimpleQuote(31)Quote));         /************************************  | Call Stack       Call Stack         Call Stack       Seekpoints         Call Stack       Breekpoints   |   |
| boost::shared_prr <quote>s5yRate(new SimpleQuote(S\$Quote));<br/>boost::shared_prr<quote>s15yRate(new SimpleQuote(s15yQuote));<br/>boost::shared_prr<quote>s15yRate(new SimpleQuote(s15yQuote));<br/>/***********************************</quote></quote></quote>   | boost:::shared_ptr <quote> s5%tate(new SimpleQuote(S5QUote));       boost:::shared_ptr<quote> s10yRate(new SimpleQuote(S10yQuote));       boost:::shared_ptr<quote> s10yRate(new SimpleQuote(S15yQuote));       /************************************</quote></quote></quote>  | boost:::shared_ptr <quote> s5%Ret(new SimpleQuote(s5%Quote));<br/>boost::shared_ptr<quote> s10%Ret(new SimpleQuote(s15%Quote));<br/>boost::shared_ptr<quote> s10%Ret(new SimpleQuote(s15%Quote));<br/>/***********************************</quote></quote></quote>  | baost:::shared_ptr <quote> s5yRate(new SimpleQuote(s5yQuote));         baost:::shared_ptr<quote> s10yRate(new SimpleQuote(s10yQuote));         baost:::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));         /************************************</quote></quote></quote>   |   |
| boost::shared_ptrQuote> si&yRate(new SimpleQuote(si&yQuote));<br>boost::shared_ptrQuote> siSyRate(new SimpleQuote(siSyQuote));         /************************************  | boost:::shared_ptr <quote> s15yRate(new SimpleQuote(s15yQuote));         boost:::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));         /************************************</quote></quote>   | boost::shared_ptr <quote> s19yRate(new SimpleQuote(s19yQuote));<br/>boost::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));<br/>/***********************************</quote></quote>   | boost::shared_ptr <quote> s18yRate(new SimpleQuote(s18yQuote));         boost::shared_ptr<quote> s15yRate(new SimpleQuote(s15yQuote));         /************************************</quote></quote>   |   |
| boost::shared_ptrQuote> s15yRate(new SimpleQuote(s15yQuote));         /************************************   | call       boost:::shared_ptr <quote> s15yRate(new SimpleQuote(s15yQuote));         /************************************</quote>  | Call Stack       Value       Type         Call Stack       Streskpoints       Output         Call Stack       Bendsynthis: Data using no symbols loaded for kernel32.dll]       Call Stack       Value         Call Stack       Type       Output       Output  | boost::shared_ptr <quote> s15yRate(new SimpleQuote(s15yQuote));         /**** name         100 % &lt; 1         Locals         Name         Value         9 s2yQuote         0.029499999999998         0.029499999999998         0.029499999999998         0.029499999999998         0.029499999999998         0.029499999999998         0.029499999999998         0.029499999999998         0.0294999999999998         0.0294999999999998         0.02949999999999998         0.0294999999999999998         0.0294999999999999998         0.029400         0.02949999999999999999999999999999999999</quote>   |   |
| Call Stack       Value       Type         Call Stack       Treads Modules       Value       Value       Value         Value       (pre-10xccccccc (use_count_=??? weak_count)       boostishared_ptr <quantlib:<br>0 actions:       boostishared_ptr<quantlib:<br>0 actions:&lt;</quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br></quantlib:<br>   | Image: State Stat  | Intervention       Intervention         100 %       Intervention <th>Image: Control of the second secon</th> | Image: Control of the second secon  |   |
| Call       **** DATE VELIGED ************************************   | Call       Value       Type         00%       1       Value       Type         00%       0.02499999999999999999999999999999999999  | Call       *** DATE USINERC ***         100 % - 4       ***         Cockis       ***         ***       Name       Value       Type         • s2yQuote       0.0294999999999999       double         • dismRate       (px=0xcccccc(-), pn=(pi=0xccccccc(use, count_=)??? weak_count_=boost:shared_ptr <quantlibuc< td="">         • dismRate       (px=0xccccccc(-), pn=(pi=0xccccccc(use, count_=)??? weak_count_=boost:shared_ptr<quantlibuc< td="">         • bondBingine       (px=0xccccccc), (use, count_=)??? weak_count_=boost:shared_ptr<quantlibuc< td="">         • bondBingine       (px=0xccccccc), (use, count_=)??? weak_count_=boost:shared_ptr<quantlibuc< td="">         • bondBingine       (px=0xcccccc), (use, count_=)??? weak_count_=boost:shared_ptr<quantlibuc< td="">         • didyState       (senalNumber_=39706)       QuantLibuc         • didyState       (senalNumber_=39706)       QuantLibuc         Call Stack       ***       ***         • Bonds-vc10-mt-gd.exelmain(int_formal, char**_formal) Line 240       Call       Call         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Type       Type         • QuantLibu:ActualActual:ISDA_Impl       shared_ptr(] [2 strong refs] [default]       boostic         • @ [pr]       0x0000bd34 [8onds-vc10-mt-gd.exelconst QuantLibu:Actual:ISDA_Impl]      </quantlibuc<></quantlibuc<></quantlibuc<></quantlibuc<></quantlibuc<>  | 100 %       **** DATE VELICERC ***         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       ****         100 %       Value         100 %       Value         100 %       Value         100 %       0.0294999999999999999999998         100 %       0.02930999999999999999999999999999999999   |   |
| 100 % - 14         Locals   | 100 % - 4       Image: Structure DayCounter: St  | 100 % - 4          Locals       Type         Name       Value       Type         4       0.0294999999999998       double         b e d3mRate       (px=0xccccccc (u) pns[pi_=0xccccccc (use_count_=??? weak_count; boost:shared_ptr <quantilis:0< td="">         c ddmRate       (px=0xccccccc () pns[pi_=0xccccccc (use_count_=??? weak_count; boost:shared_ptr<quantilis:0< td="">         c ddmRate       (px=0xccccccc () pns[pi_=0xccccccc (use_count_=??? weak_count; boost:shared_ptr<quantilis:0< td="">         c ddmSate       (px=0xcccccccc () pns[pi_=0xccccccc (use_count_=??? weak_count; boost:shared_ptr<quantilis:0< td="">         b e bondEngine       (px=0xccccccc (stag_count_=??? weak_count; boost:shared_ptr<quantilis:0< td="">         c cals       (tro = new 0xccccccccc (stag_count_=??? weak_count; boost:shared_ptr<quantilis:0< td="">         b e bondEngine       (px=0xcccccccc (stag_count_=??? weak_count; boost:shared_ptr<quantilis:0< td="">         c cals       (tro = new 0xcccccccccccccccccccccccccccccccccccc</quantilis:0<></quantilis:0<></quantilis:0<></quantilis:0<></quantilis:0<></quantilis:0<></quantilis:0<>  | 100 % • 4       • 1         Locals       • 1         Name       Value       Type         • \$2yQuote       0.0294999999999998       double         • • d3mRate       (px=0xcccccc() pn=(pi_=0xccccccc(use_count_=??? weak_count_! boost::shared_ptr <quantlib::q< td="">         • • d1yRate       (px=0xccccccc() pn=(pi_=0xccccccc(use_count_=??? weak_count_! boost::shared_ptr<quantlib::q< td="">         • • d1yRate       (px=0xccccccc() pn=(pi_=0xccccccc(use_count_=??? weak_count_! boost::shared_ptr<quantlib::q< td="">         • • bondEngine       {px=0xcccccccc() pn=(pi_=0xcccccccc(use_count_=??? weak_count_! boost::shared_ptr<quantlib::d< td="">         • • totoaysDate       {px=0xcccccccc() pn=(pi_=0xcccccccc(use_count_=??? weak_count_! boost::shared_ptr<quantlib::d< td="">         • • totoaysDate       {px=0xcccccccc() pn=(pi_=0xccccccccc(use_count_=??? weak_count_! boost::shared_ptr<quantlib::d< td="">         • • totoaysDate       {px=0xccccccccc() pn=(pi_=0xccccccccc(use_count_=??? weak_count_! boost::shared</quantlib::d<></quantlib::d<></quantlib::d<></quantlib::d<></quantlib::d<></quantlib::d<></quantlib::d<></quantlib::q<></quantlib::q<></quantlib::q<>   |   |
| Locals       Value       Type         Name       Value       Type         ● s2yQuote       0.024499999999998       double         >● ddmRate       (px=Dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx   | Locals       value       Type         Name       Value       Type         ● d3mRate       (preDxccccccc(u) pn=[pi_=0xccccccc(us_count_=??? weak_count_       boost:shared_ptr <quantlib:s< td="">         ● d1yRate       (preDxcccccccc(u) pn=[pi_=0xccccccc(us_count_=??? weak_count_       boost:shared_ptr<quantlib:s< td="">         ● dodsplate       (preDxcccccccc(u) pn=[pi_=0xccccccc(us_count_=??? weak_count_       boost:shared_ptr<quantlib:s< td="">         ● dodsplate       (preDxcccccccc(u) pn=[pi_=0xccccccc(us_count_=??? weak_count_       boost:shared_ptr<quantlib:s< td="">         ● dodsplate       (scrallNumber_=39706)       QuantLib:Date         ● dodsplate       (scrallNumber_=39706)       QuantLib:Date         Call Stack       •       Count_dots       Count_dots         Name       [Locals]       Threads       Modules       Watch1         Call Stack       •       Cods       Code       [Citateral Code]         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       •       Type         • for [ptr]       shared_ptr () [2 strong refs] [default] )       Quant       Quant         • [ptr]       shared_ptr () [2 strong refs] [default] )       Quant       Quant         • [ptr]       shared_ptr () [2 strong refs] [default] )       Quant       Quant</quantlib:s<></quantlib:s<></quantlib:s<></quantlib:s<>  | Locals       Value       Type         Name       Value       Type         ● ● d3mRate       (px=0xccccccc() pn=[pi=0xcccccc (use_count_=??? weak_count: boost::shared_ptr <quantlib::q< td="">         ● ● d3mRate       (px=0xcccccccc() pn=[pi=0xcccccc (use_count_=??? weak_count: boost::shared_ptr<quantlib::q< td="">         ● ● dayRate       (px=0xcccccccc() pn=[pi=0xcccccc (use_count_=??? weak_count_: boost::shared_ptr<quantlib::q< td="">         ● ● bondEngine       (px=0xcccccccc() pn=[pi=0xccccccc (use_count_=??? weak_count_: boost::shared_ptr<quantlib::q< td="">         ● ● bondStpate       (EeraiRhumber_3706)       QuantLib::D         ● ● todayRate       (eraiRhumber_3706)       QuantLib::Q         Name       (eraiRhumber_3706)       QuantLib::Q         Name       [a       Bonds-vc110-mt-gd.exelmain(int_formal, char**_formal) Line 240       C+         [x+rmal Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       QuantLib::D       QuantLib::Q         Call Stack       Breakpoints       Output       Shared_ptr () [2 strong refs] [default])       QuantLib::Q         Call Stack       Implshared_ptr () [2 strong refs] [default]       boost::hared_ptr () [2 strong refs] [default]       Doost::hared_ptr () [2 strong refs] [default]       QuantLib::D         • ● [ptr]      </quantlib::q<></quantlib::q<></quantlib::q<></quantlib::q<>  | Locals       Value       Type            • s2yQuote         • 0.02949999999999998         • d3mRate         {px=0xccccccc {}pn={pi_=0xccccccc {use_count_=??? weak_count; boost::shared_ptr <quantlib::q -9.2559631349317831e+061="" <="" bondengine="" boost::shared_ptr<quantlib::q="" cerror="" characters="" double="" e="" faceamount="" of="" reading="" separator="" string.="" weak_count;="" {px="0xccccccc" {use_count_="???" {}pn="{pi_=0xccccccc" •="">         Q &lt; std::basic_string<char,std::chare <="" p="" td="">            • odaysDate          (px=0xcccccccc {}pn={pi_=0xcccccccc {use_count_=??? weak_count; boost::shared_ptr<quantlib::p <="" separator="" td="" •="">            • dodaysDate          (px=0xcccccccc {}pn={pi_=0xcccccccc {use_count_=??? weak_count; boost::shared_ptr<quantlib::p <="" dodaysdate="" td="" •="">            • dodaysDate          (px=0xcccccccc {}pn={pi_=0xcccccccc {use_count_=??? weak_count; boost::shared_ptr<quantlib::p <="" td="">            • dodaysDate          (px=0xcccccccc {use_count_=??</quantlib::p></quantlib::p></quantlib::p></quantlib::p></quantlib::p></quantlib::p></quantlib::p></char,std::chare></quantlib::q>  |   |
| Instance     Particle     Particle     Particle <ul> <li>             displace</li> <li>             displace</li> </ul> <ul>             displace             displace </ul> <ul>             displace             displace&lt;</ul>   | Instruction       Value       Instruction            • SigQuote        0.0299999999999999999999999999999999999   | Call Stack       Value       Type         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output   | Internet       Value       Itype            • s2yQuote         • 0.02949999999999999         • 0.02949999999999999999999999999999999999  |   |
| ScyQuote     Output     ScyQuote     Output     O  | • stycuote       UU2499999999999999999999999999999999999   | • e.g. 2000te       00.2049999999999999999999999999999999999  | • szyQuote       0.0249999999999999       double         • • d3mRate       {px=0xccccccc() pn={pi_=0xccccccc {use_count_=??? weak_count_} is boost::shared_ptr <quantlib::q< td="">         • • d1yRate       {px=0xcccccccc() pn={pi_=0xcccccccc {use_count_=??? weak_count_} is boost::shared_ptr<quantlib::q< td="">         • • bondEngine       {px=0xcccccccc() pn={pi_=0xcccccccc {use_count_=??? weak_count_} is boost::shared_ptr<quantlib::p< td="">         • • separator       <error characters="" of="" reading="" string.="">       Q + std::basic_string<char,std::char< td="">         • • todaysDate       {serialNumber_=39706}       QuantLib::Date         • • and       0.0030/30/3018 (fuice_Date_formal, char **_formal) Line 240       C+         • Bonds-vc110-mt-gd.exe!main(int_formal, char **_formal) Line 240       C+         • [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       C+</char,std::char<></error></quantlib::p<></quantlib::q<></quantlib::q<>   |   |
| V       Obstrate       (producted (iii) projected (iii) projected (iii) producted (iiiii) producted (iii) producted (iii) producted (iiii) pr   | ✓ dofinate       (paroAccecccc (, phep)=putcecccc (use_count_=??? weak_count_       boosts:shared_ptr       putceccccc (, phep)==vaccecccc (use_count_=??? weak_count_       boosts:shared_ptr       double         ♦ of doffspine       (px=0xccccccc (, phep)==0xccccccc (use_count_=??? weak_count_       boosts:shared_ptr       double         ♦ of doffspine       (px=0xccccccc (, phep)==0xccccccc (use_count_=??? weak_count_       boosts:shared_ptr       quantilib:         ♦ of doffspine       (px=0xccccccc (, phep)==0xcccccccc (use_count_=??? weak_count_       boosts:shared_ptr       quantilib:         ♦ of todaysDate       (serialNumber_=39706)       Quantilib::Date       (serialNumber_=39706)       Quantilib::Date         Call Stack       *       *       *       *       *       *       Call Stack       *         Name  | V Burnate       Up=0xttttttt (, pin=(pi_=0xtttttttt (, pin=(pi_=0xttttttttttt))       boots.shafed_pin       guantilise         V b diffyRise       [pi=0xttttttttt] [pi=0xtttttttttt] boots.shafed_pin       boots.shafed_pin       guantilise         V b bondEngine       [pi=0xttttstt] boots.shafed_pin       boots.shafed_pin       guantilise         V b bondEngine       [pi=0xttttstt] boots.shafed_pin       guantilise         V b bondEngine       [pi=0xttttst] boots.shafed_pin       guantilise         V b bondEngine       [pi=0xtttst] boots.shafed_pin       guantilise         V b bondEngine       [pi=0xtttst] boots.shafed_pin       guantilise         V b bondEngine       [pi=0xttst] boots.shafed_pin       guantilise         V boots       [coals block.shafed_pin       [pi=0xttst] boots.shafed_pin       guantilise         Call Stack       v is       [pi=0xttst] boots.shafed_pin       [pi=0xttst] boots.shafed_pin       guantilise         Call Stack       Breakpoints       Output       [pi=0xttst] boots.shafed_pin       [pi=0xttst] boots.shafed_pin       [pi=0xttst] boots.shafed_pin       [pi=0   | V       Ostimate       (pr=0xccccccc (us) pn=(pi=0xccccccc (use_count_=??? weak_count_) boost:shared_ptr <quantlib:qu< td=""></quantlib:qu<> |   |
| Call Stack       Impacts (in program (  | Second Constrained Difference Constraine  | Grackmount           Grackmount </td <td><ul> <li></li></ul></td>   | <ul> <li></li></ul>  |   |
| > bondfignine       (px=0xccccccc () pn=(pi_=0xccccccc (use_count_=??? weak_count; boost:shead_ptr <quantlib::<br>boost:shead_ptr<quantlib::<br>&gt; cloadsyDate       (px=0xccccccc () pn=(pi_=0xccccccc (use_count_=??? weak_count; boost:shead_ptr<quantlib::<br>&gt; Q + std:basic_string         &gt; CodaysDate       (px=0xccccccc () pn=(pi_=0xccccccc (use_count_=??? weak_count; boost:shead_ptr<quantlib::date< td="">         &gt; CodaysDate       (px=0xcccccccc () pn=(pi_=0xcccccccc (use_count_=??? weak_count; boost:shead_ptr<quantlib::date< td="">         &gt; CodaysDate       (px=0xccccccccc)         Call Stack       (px=0xcccccccc)         Name       L         Bonds-vc110-mt-gd.exelmain(int_formal, char ** _formal) Line 240       L         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Type         @ temStructureDayCounter       (impl_=shared_ptr {) [2 strong refs] [default] &gt;       QuantLib::ActualActual:ISDA_Impl]         &lt; @ gimpl_</quantlib::date<></quantlib::date<></quantlib::<br></quantlib::<br></quantlib::<br>   | Call Stack       Value       Type         Call Stack       Value       Value       Call Stack       Value         Call Stack       Value       Value       Call Stack       Value       Call Stack         Call Stack       Value       Value       Call Stack       Value       Call Stack       Value         Call Stack       Value       Value       Call Stack       Value       Call Stack       Value         Call Stack       Value       Call Stack       Value       Call Stack       Value       Call Stack       Value       Call Stack       Value       Call Stack       Value       Call Stack       Call Stack       Value       Call Stack       Value       Call Stack       Call Stack       Value       Call Stack   | Image: Second   | Call Stack       Call Stack         Call Stack       Call Stack         Call Stack       Call Stack         Call Stack       Call Stack  |   |
| > • separator    > • separator    > • construction    > • construction <td>Separator     Carror reading characters of string.&gt;     Q - std::basic_string-characters of string.&gt;     QuantLib::Date     Coals Threads Modules Watch 1     Call Stack     Name     Quant Lib::Date     Call Stack     Reakpoints Output  Call Stack     Reakpoints Output  Call Stack     Value     Value     Value     Type     (impl_=shared_ptr {}] 2 strong refs] [default] &gt;     Quant     Quant</td> <td>image: separator       Crror reading chary pupper separator       Quantilibility of the separator         image: separator       Crror reading chary pupper separator       Quantilibility of the separator         image: separator       Crror reading chary pupper separator       Quantilibility of the separator         image: separator       Crror reading chary pupper separator       Quantilibility of the separator         image: separator       Crror reading chary pupper separator       Quantilibility of the separator         Autos       Locals       Threads       Modules         Autos       Locals       Threads       Modules         Separator       Crror reading chary pupper separator       Quantilibility of the separator         Call Stack       Code       Line       Line         Pandes-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       Code       Code         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack       Type         Call Stack       Breakpoints       Output       Value       Type         iame       Value       Value       Type       Type         @ ternsStructureDayCounter       [impl_=shared_ptr { } [2 strong refs] [default]       boots:       Quantilib: ActualActual::ISDA_Impl]       Quantit         i @ [ptr]</td> <td>Image: Separator       Crror reading characters of string.&gt;       QuantLib:Date         Image: Call Stack       Call Stack       Image: Call Stack         Image: Call Stack       Call Stack       Call Stack         Stack       Stack       Call Stack         Call Stack       Call Stack       Call Stack         Stack       Stack       Call Stack         Call Stack       Stack       Call Stack         Stack       Stack       Call Stack         Stack       Stack       Stack         Stack       Stack       Stack         Stack       Stack       Stack         St</td>   | Separator     Carror reading characters of string.>     Q - std::basic_string-characters of string.>     QuantLib::Date     Coals Threads Modules Watch 1     Call Stack     Name     Quant Lib::Date     Call Stack     Reakpoints Output  Call Stack     Reakpoints Output  Call Stack     Value     Value     Value     Type     (impl_=shared_ptr {}] 2 strong refs] [default] >     Quant     Quant   | image: separator       Crror reading chary pupper separator       Quantilibility of the separator         image: separator       Crror reading chary pupper separator       Quantilibility of the separator         image: separator       Crror reading chary pupper separator       Quantilibility of the separator         image: separator       Crror reading chary pupper separator       Quantilibility of the separator         image: separator       Crror reading chary pupper separator       Quantilibility of the separator         Autos       Locals       Threads       Modules         Autos       Locals       Threads       Modules         Separator       Crror reading chary pupper separator       Quantilibility of the separator         Call Stack       Code       Line       Line         Pandes-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       Code       Code         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack       Type         Call Stack       Breakpoints       Output       Value       Type         iame       Value       Value       Type       Type         @ ternsStructureDayCounter       [impl_=shared_ptr { } [2 strong refs] [default]       boots:       Quantilib: ActualActual::ISDA_Impl]       Quantit         i @ [ptr]  | Image: Separator       Crror reading characters of string.>       QuantLib:Date         Image: Call Stack       Call Stack       Image: Call Stack         Image: Call Stack       Call Stack       Call Stack         Stack       Stack       Call Stack         Call Stack       Call Stack       Call Stack         Stack       Stack       Call Stack         Call Stack       Stack       Call Stack         Stack       Stack       Call Stack         Stack       Stack       Stack         Stack       Stack       Stack         Stack       Stack       Stack         St  |   |
| b @ todaysDate       [serialNumber_=39706 )       QuantLib::Date         Autos Locals       Threads       Modules       Watch 1         Call Stack       Image: Control of the Control o   | Image: State (serialNumber_=39706)       QuantLibr:Date (serialNumber_=39706)       QuantLibr:Date (serialNumber_=30706)         Autors       Locals       Threads       Modules       Watch 1         Call Stack       •       •       0.0070/d318 (finiseDate (serialNumber_=30706) libradeder (serialNu   | <ul> <li>interaction of the second second</li></ul>  | Image: Control of the second secon  |   |
| Call Stack       Implementation         Call Stack       Implementation         O Bonds-vc110-mt-gd.exelmain(int_formal, char**_formal) Line 240       Implementation         Call Stack       Implementation         O termStructureDayCounter       (impl_=shared_ptr () [2 strong refs] [default] )         O QuantLib::ActualActual:ISDA_Impl]       ()         Implementation       ()         Implementation       ()         Implementation       ()         Implementation  | Autos       Locals       Threads       Modules       Watch1         Call Stack <td>Autos       Locals       Threads       Modules       Watch 1         Call Stack        ////////////////////////////////////</td> <td>Autos       Locals       Threads       Modules       Watch 1         Call Stack        Image: Call Stack        Image: Call Stack         Name        La         Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]           [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]</td>  | Autos       Locals       Threads       Modules       Watch 1         Call Stack        ////////////////////////////////////   | Autos       Locals       Threads       Modules       Watch 1         Call Stack        Image: Call Stack        Image: Call Stack         Name        La         Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]           [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]   |   |
| Call Stack   Name   Bonds-vcl10-mt-gd.exelmain(int _formal, char ** _formal) Line 240   [External Code]   [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]   Call Stack Breakpoints Output Cals Type Cals Type Cals Cals Cals Cals Cals Cals Cals Cals  | Call Stack    Name Li   Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240 C   [External Code] C   [Errames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]     Call Stack   Breakpoints   Output     Call Stack     Breakpoints   Output     Call Stack     Breakpoints   Output     Call Stack     Breakpoints   Output     Call Stack     Breakpoints   Output     Call Stack     Breakpoints   Output     Call Stack   Breakpoints   Output     Call Stack     Breakpoints   Output     Call Stack     Breakpoints   Output     Call Stack     Breakpoints   Output     Call Stack     Breakpoints   Output     Value   Image: Stack  | Call Stack       Ia         Name       La         Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]       C+         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       C         Call Stack       Breakpoints       Output         Image       Value       Type   | Call Stack          Call Stack       La         O Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]       C+         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       C         Call Stack       Breakpoints       Output   |   |
| Call Stack       Image: Stack of the stack   | Call Stack       IL         Name       Li         Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       C         Call Stack       Breakpoints       Output         Call Stack  | Call Stack       Ia         Name       Ia         Image: Stack       Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack         Breakpoints       Output       Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output       Type         Call Stack       Breakpoints       Output       Value       Type         Image: Value       Value       Value       Type       Quanti   | Call Stack       La         Name       La         Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints       Output  |   |
| Name     L          Bonds-vc110-mt-gd.exelmain(int_formal, char ** _formal) Line 240       [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack Breakpoints Output       Call Stack Value       Valu   | Name       Li         © Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C         [External Code]       [External Code]         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack         Breakpoints       Output         *         Call Stack         Breakpoints       Output         *         Call Stack         Breakpoints         Value         * <td colspa<="" td=""><td>Name       La         © Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack         Call Stack       Breakpoints       Output         Value       Type         Output       Value       Type       Quantl         Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl       Quantl         Impl</td><td>Name       La         Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints       Output</td></td>   | <td>Name       La         © Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack         Call Stack       Breakpoints       Output         Value       Type         Output       Value       Type       Quantl         Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl       Quantl         Impl</td> <td>Name       La         Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints       Output</td>   | Name       La         © Bonds-vc110-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack         Call Stack       Breakpoints       Output         Value       Type         Output       Value       Type       Quantl         Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl       Quantl         Impl  | Name       La         Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints       Output |
| Bonds-vc110-mt-gd.exelmain(int_formal, char**_formal) Line 240       C         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack         Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Breakpoints       Output         Call Stack       Impl   | Object       Bonds-vcl10-mt-gd.exelmain(int _formal, char ** _ formal) Line 240       C         [External Code]       [External Code]       [External Code]         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]       Call Stack       Breakpoints       Output         Value       Type         Quant         Quant         Quant       Output       Stared_ptr {} [2 strong refs] [default]       Dootst         Quant       Output       Output       Quant         QuantLibs::ActualActual::ISDA_Impl]  | Bonds-vcl10-mt-gd.exelmain(int _formal, char ** _formal) Line 240       C+         [External Code]       [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints       Output         Call Stack       Type         August colspan="2">Call Stack       Type         August colspan="2">Call Stack       Type         Call Stack       Type         Call Stack       Type         Output         Start Colspan="2">Call Stack       Type         Call Stack<  | Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240     [External Code]     [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]  Call Stack Breakpoints Output  |   |
| [External Code]         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints         Output         cals         vame       Value         Value       Type         Image: Stack StructureDayCounter       {impl_shared_ptr {} {2 strong refs] [default] }         Quant Image: Stack StructureDayCounter       (impl_shared_ptr {} {2 strong refs] [default] }         Quant Image: Stack StructureDayCounter: Stared ptr {} {2 strong refs] [default] }       Boost         Image: QuantLib::ActualActual::ISDA_Impl]       {}         Quant Image: QuantLib::DayCounter::Impl       {}         Quant Image:  | [External Code]         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints         Output         cals         vame       Value         Impl_=shared_ptr {}[2 strong refs] [default]}         Quant         Impl_         Shared_ptr {}[2 strong refs] [default]         Impl_         Shared_ptr {}[2 strong refs] [default]         Quant         Impl_         Ox0082ff88 {}         QuantLib::ActualActual:ISDA_Impl]         Impl_         Ox0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual:ISDA_Impl::Vrtabl void *         Impl_       Ox0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual:ISDA_Impl::Vrtabl void *         Impl_       Ox0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual:ISDA_Impl::Vrtabl void *         Impl_       Ox0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual:ISDA_Impl::Vrtabl void *         Impl_       Ox0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual:ISDA_Impl::Vrtabl void *         Impl_       Ox0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual:ISDA_Impl::Vrtabl void *         Impl_       Ox0082ff88 {}       Ox0082ff88 {}   | [External Code]         [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints         Output         cals         vame       Value         Impl_shared_ptr {}[2 strong refs] [default] }         Quanti         Impl_         shared_ptr {}[2 strong refs] [default]         Impl_         Shared_ptr {  | [External Code]<br>[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll] Call Stack Breakpoints Output  |   |
| [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints         Output         cals         value       Type         Image: Stack       Value   | [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints       Output         Image: Stack       Call Stack       Breakpoints       Output         Image: Stack       Value       Type         Image: Stack       Value       Type         Image: Stack       Call Stack       Type         Image: Stack       Value       Type         Image: Stack       Stared_ptr {}[2 strong refs] [default] }       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Quant       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Doost       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Quant       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Quant       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Quant       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Quant       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Quant       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Quant       Quant         Image: Stared_ptr {}[2 strong refs] [default] }       Quant       Quant         Image: Stared_ptr {}[2 strong refs] [defa   | [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]         Call Stack       Breakpoints       Output         icals       Value       Type         Name       Value       Type         Image: Image   | [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll] Call Stack Breakpoints Output   |   |
| Name     Value     Type       Impl_eshared_ptr {} [2 strong refs] [default] }     Quan       Impl_eshared_ptr {} [2 strong refs] [default] }     boost       Impl_eshared_ptr {} [2 strong refs] [default] }     Quan       Impl_eshared_ptr {} ?     QuantLib::DayCounter::Impl     Quan       Impl_eshared_ptr {} ?     QuantLib::ActualActual::ISDA_Impl::Mtabl     Void %  | Decals       Value       Type         Name       Value       Type         Image: ConstructureDayCounter       {impl_=shared_ptr {} [2 strong refs] [default] }       Quant         Image: ConstructureDayCounter       {impl_=shared_ptr {} [2 strong refs] [default] }       Quant         Image: ConstructureDayCounter       {impl_=shared_ptr {} [2 strong refs] [default] }       Dootst         Image: ConstructureDayCounter       Quant       Dootst       Quant         Image: ConstructureDayCounter::Impl       {}       Quant       Quant         Image: ConstructureDayCounter::Impl       {} <th< th=""><th>Decals       Value       Type         Name       Value       Type         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptrefaulter) (image: Constructurer       Quanti</th><th></th></th<> | Decals       Value       Type         Name       Value       Type         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptr {}] [2 strong refs] [default] }       Quantility         Image: ConstructureDayCounter       (impl_=shared_ptrefaulter) (image: Constructurer       Quanti  |  |   |
| Name     Value     Type       Impl_shared_ptr {}[2 strong refs] [default] }     Quant       Impl_shared_ptr {}[2 strong refs] [default] }     boost       Impl_shared_ptr {}[2 strong refs] [default] }     Quant       Impl_shared_ptr {}[2 strong refs] [default] }     Quant       Impl_shared_ptr {}[2 strong refs] [default] }     Boost       Impl_shared_ptr {}[2 strong refs] [default] }     Quant       Impl_shared_ptr []     Quant     Quant       Impl_shared_ptr []     Quant     Quant       Impl_shared_ptrefault     Quant     Q   | Name       Value       Type         Image: StructureDayCounter       {impl_=shared_ptr {} {2 strong refs] [default] }       Quant         Image: Shared_ptr {}       {impl_eshared_ptr {} {2 strong refs] [default] }       boost         Image: Shared_ptr {}       Quant       \$wared_ptr {} {2 strong refs] [default] }       boost         Image: Shared_ptr {}       Quant       Quant       Quant       Quant         Image: Shared_ptr {}       Quant       Quant       Quant       Quant       Quant         Image: Shared_ptr {}       Quant       Quant       Quant       Quant       Quant         Image: Shared_ptr {}       Quant       Quant       Quant       Quant       Quant         Image: Shared_ptr {}       Image: Shared_ptr {}       Quant       Quant       Quant   | Value     Type       Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     Doots:       Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {  |  |   |
| Implementation       Value       (impl_=shared_ptr {}] 2 strong refs] [default] }       Quant         Implementation       shared_ptr {}] 2 strong refs] [default] }       boost         Implementation       shared_ptr {}] 2 strong refs] [default] }       boost         Implementation       shared_ptr {}] 2 strong refs] [default] }       boost         Implementation       wide       wide       Quant         Implementation       wide       Quant       Quant         Implementation       wide       Quant       Quant         Implementation       Quant       Quant       Quant       Quant         Implementation       Quant       Quant       Quant       Quant       Quant         Implementation       Quant       Quant   | Value       Value       Type         Value       {impl_=shared_ptr {} [2 strong refs] [default] }       Quant         V Impl_       shared_ptr {} [2 strong refs] [default] }       boost         V Impl_       shared_ptr {} [2 strong refs] [default] }       boost         V Impl_       shared_ptr {} [2 strong refs] [default] }       boost         V Impl_       0x0082ff88 {}       Quant         V Impl_       0x0082ff88 {}       Quant         V Impl_       QuantLib::ActualActual::ISDA_Impl]       {}         V Impl_       QuantLib::DayCounter::Impl       {}       Quant         V Impl_       QuantLib::DayCounter::Impl       {}       Quant         V Impl_       QuantLib::DayCounter::Impl       {}       Quant         V Impl_       Quant       Quant       Quant       Quant       Quant <th>Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Impl</th> <th>Cals Value Ture</th>   | Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Impl  | Cals Value Ture  |   |
| ▲ ♀ impl_       shared_ptr {} [2 strong refs] [default]       boost         ▲ ♀ [ptr]       0x0082ff88 {}       Quan         ▲ ♀ [QuantLib::ActualActual:ISDA_Impl]       {}       Quan         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quan         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quan         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quan         ▶ ♀ _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual:ISDA_Impl::\ftabl       void <sup>4</sup> ▶ ♀ [deleter and allocator]       default       boost         Locals       Threads       Modules       Watch 1   | ▲ ♥ impl_       shared_ptr {} [2 strong refs] [default]       boost         ▲ ● [ptr]       0x0082ff88 {}       Quant         ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▶ ● QuantLib::DayCounter::Impl       {}       Quant         ▶ ●   | ▲ ♥: impl_       shared_ptr {} [2 strong refs] [default]       boost::         ▲ ● [ptr]       0x0082ff88 {}       Quantl         ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▶ ● QuantLib::DayCounter::Impl       {}       Quantl         ▶ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\vftabl void * ``         ▶ ● [deleter and allocator]       default       boost::         ▶ ● [fourle_rand       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\vftabl void * ``  |  |   |
| ▲ ● [ptr]       0x0082ff88 {}       Quan         ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quan         ▶ ● QuantLib::DayCounter::Impl       {}       Quan         ▶ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::\ftabl       Quan         ▶ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::\ftabl       Void 7         ▶ ● [deleter and allocator]       default       boost         tos       Locals       Threads       Modules       Watch 1         Il Stack   | ▲ ● [ptr]       0x0082ff88 {}       Quant         ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▶ ● QuantLib::DayCounter::Impl       {}       Quant         ▶ ● QuantLib::DayCounter::Impl       {}       Quant         ▶ ● QuantLib::DayCounter::Impl       {}       Quant         ▶ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::`vftabl void *         ▶ ● [deleter and allocator]       default       boost         Locals       Threads       Modules       Watch 1   | ▲ ● [ptr]       0x0082ff88 {}       Quantl         ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷ ● QuantLib::DayCounter::Impl       {}       Quantl         ▷ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``         ▷ ● [deleter and allocator]       default       boost:         ▷ ● [Ibrue]       0x0060bd570 () 0x0060bf34 {bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``  | Gree      Figure     (impl = shared ptr {}[2 strong refs] [default] }     Quantl   |   |
|   | ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▷ ● QuantLib::DayCounter::Impl       {}       Quant         ▷ ● QuantLib::DayCounter::Impl       {}       Quant         ▷ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::`vftabl void *         ▷ ● [deleter and allocator]       default       boost         Locals       Threads Modules Watch 1       0x00126570 (cm. 0x00824688 (c) cm. (ci. 0x00821178 default ))       boost   | ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       QuantLib::DayCounter::Impl       {}         ▶ ● QuantLib::DayCounter::Impl       {}       QuantLib::ActualActual::ISDA_Impl::Vftabl       QuantLib::ActualActual::ISDA_Impl::Vftabl         ▶ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl       void * ``         ▶ ● [deleter and allocator]       default       boost:         ▶ ● [Impl://sml       0x005670 (m) 0x0050478 (m) model/mainter       boost:  | Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       shared_ptr {} [2 strong refs] [default] }       boost:   |   |
| <ul> <li>QuantLib::DayCounter::Impl</li> <li>QuantLib::DayCounter::Impl</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void <sup>7</sup></li> <li>QuantLib::ActualActualActual::ISDA_Impl::Vftabl void <sup>7</sup></li> <li>QuantLib::ActualAct</li></ul>  | ▷ ● QuantLib::DayCounter::Impl       {}       Quant         ▷ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::`vftabl void *         ▷ ● [deleter and allocator]       default       boost         tos Locals       Threads Modules Watch 1       0x00126=70 (cm. 0x00824688 (c) cm. (ci. 0x00821178 default ))       boost  | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib::         ▷ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * '         ▷ ● [deleter and allocator]       default         ▷ ● [deleter and allocator]       0x0060bd72 (L) and (L   | Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Implshared_ptr {} [2 strong refs] [default] }       Quantl         Impl       shared_ptr {} [2 strong refs] [default] }       boost::         Impl       0x0082ff88 {}       Quantl   |   |
| <ul> <li>vfptr</li> <li>0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \ftabl void '</li> <li>default</li> <li>default</li> <li>boost</li> <li>Locals</li> <li>Threads</li> <li>Modules</li> <li>Watch 1</li> </ul>  | > •  | ▶ ●vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * 1         ▶ ● [deleter and allocator]       default         ▶ ● [flow]/(m)       0x006570 (m) 0x005200 (c) how (m) 0x0021170 (c) how   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] ]       [1 strong refs] [default] ]   |   |
| b @ [deleter and allocator]     default     boost       tos     Locals     Threads     Modules       Warre     Name     Il Stack  | ▷ ● [deleter and allocator]     default     boost       tos     Locals     Threads     Modules     Watch 1   | ▷ ● [deleter and allocator] default boost:  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         I  |   |
| tos Locals Threads Modules Watch 1  | tos Locals Threads Modules Watch 1   | b   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Doots:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] ]       Quantl         I  |   |
| II Stack  |  | tos Locals Threads Modules Watch1   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         I  |   |
| Name  |  |   | impl_shared_ptr {} [2 strong refs] [default] }       Quantl         impl_simpl_       shared_ptr {} [2 strong refs] [default] }       Quantl         impl_simpl_       shared_ptr {} [2 strong refs] [default] }       boost:         impl_simpl_       0x0082ff88 {}       Quantl         impl_simpl_   |   |
|   |  | Stack   | Impl_       impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       shared_ptr {} [2 strong refs] [default] }       boost:         Impl_       shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       shared_ptr {} [2 strong refs] [default]        Quantl         Impl_       strong refs] [default]        Quantl         Impl_       strong refs] [default]        Strong refs] [default]        Quantl         Impl_       strong refs] [default]        Strong refs] [default]        Strong refs]          Impl_       Strong refs] [default]        Strong refs] [default]        Strong refs]          Impl_   |   |
| Bonds-vc110-mt-od.exelmain(intformal_char **formal) Line 240  | Name 12  | Name  | Impl       Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} {       Quantl         Impl_shared_ptr       Quantl         Impl_shar  |   |
| [External Code]   | Name     La       Bonds-vc110-mt-od.exelmain(int_formal, char**_formal) Line 240     C.  | Name Bonds-vc110-mt-od.exelmain(int_formal, char **_formal) Line 240 C+   | Impl       Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} Quantl       Quantl <t< td=""></t<>   |   |
| L   | Name       La         Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240       C-         [External Code]       C-  | Name Lai<br>Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240 C+  | Impl       Impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       shared_ptr {} [2 strong refs] [default] }       Doots:         Impl_       shared_ptr {} [2 strong refs] [default] }       Doots:         Impl_       shared_ptr {} [2 strong refs] [default] }       Doots:         Impl_       Shared_ptr {} [2 strong refs] [default] }       Doots:         Impl_       Shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       Shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       Shared_ptr {} {       Quantl         Impl_       Shared_ptr {} {       Quantl         Impl_       Quantl       Quantl       Quantl         Impl_       Quantl       Quantl       Quantl         Impl_       Quantl <td< td=""></td<>  |   |
| [Frames below may be incorrect and/or missing, no symbols loaded for kernel22 dll]  | Name     La       Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240     C-       [External Code]     [Erames below may be incorrect and/or missing, no symbols loaded for kernel32 dll]  | Name     Lai       Bonds-vc110-mt-gd.exe!main(int _formal, char ** _formal) Line 240     C+       [External Code]     [External Code]   | Implshared_ptr {} [2 strong refs] [default] }       Quantl <ul> <li></li></ul>   |   |
| Bonds-vc110-mt-gd.exe!main(int _formal, char * * _formal) Line 240  | Call Stack   | utos <mark>Locals</mark> Threads Modules Watch1   | Image: StructureDayCounter       {impl_=shared_ptr {} [2 strong refs] [default] }       Qu         Image: Shared_ptr {} [2 strong refs] [default] }       bo         Image: Shared_ptr {} [2 strong refs] [default] }       bo         Image: Shared_ptr {} [2 strong refs] [default] }       bo         Image: Shared_ptr {} [2 strong refs] [default] }       bo         Image: Shared_ptr {} [2 strong refs] [default] }       bo         Image: Shared_ptr {} [2 strong refs] [default] }       bo         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] }       cu         Image: Shared_ptr {} [2 strong refs] [default] ]       cu         <   |   |
| I Stack   |  | tos Locals Threads Modules Watch1   | impl_       termStructureDayCounter       {impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         ✓        impl_       shared_ptr {} [2 strong refs] [default] }       boost:         ✓        [ptr]       0x0082ff88 {}       Quantl         ✓        [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷        QuantLib::DayCounter::Impl       {}       Quantl         ▷        _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::\vftabl       void **         ▷        [deleter and allocator]       default       boost:       0x0082ff28 () ar       (r) 0x082ff28 () ar   |   |
| Il Stack  |  | tos <mark>Locals</mark> Threads Modules Watch1  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         I  |   |
| Il Stack  |  | tos Locals Threads Modules Watch1   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         I  |   |
| Il Stack  | tos cours micaus modules watchi  | tos Locals Threads Modules Watch 1  | Implshared_ptr {} [2 strong refs] [default] }       Quantl         Implshared_ptr {} [2 strong refs] [default] }       Quantl         Implshared_ptr {} [2 strong refs] [default] }       Doots:         Implshared_ptr {} [2 strong refs] [default] }       Quantl         Implshared_ptr {} {       Quantl  |   |
| Il Stack  | tos Locais miedos modules watch I  | too Locals Theorem Medules Wetch 1  | Implshared_ptr {} [2 strong refs] [default] }       Quantl         Implshared_ptr {} [2 strong refs] [default] }       Quantl         Implshared_ptr {} [2 strong refs] [default] }       Doots:         Implshared_ptr {} [2 strong refs] [default] }       Quantl         Implshared_ptr {} {       Quantl  |   |
| I Stack   | tos Locais Threads Modules Watch I   | Locale Threeds Madula Week1   | Impl   |   |
| I Stack   | tos Locais Threads Modules Watch I   | Logic Threeds Madula Webba  | Impl_       Shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       Shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       Shared_ptr {} [2 strong refs] [default] }       boost:         Impl_       Shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       Shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       Shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_       Impl_       Quantl       Quantl         Impl_       Impl_       Quantl       Quantl         Impl_       Impl_       Impl_       Quantl         Impl_       Impl_       Impl_       Quantl         Impl_       Impl_       Impl_       Quantl         Impl_       Impl_       Impl_       Impl_         Impl_       Impl_       <   |   |
| I Stack   | tos Locals Threads Modules Watch1  |   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         ✓ endote       shared_ptr {} [2 strong refs] [default] }       Quantl         ✓ endote       shared_ptr {} [2 strong refs] [default] }       boost:         ✓ endote       [ptr]       0x0082ff88 {}       Quantl         ✓ endote       QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷ endote       QuantLib::DayCounter::Impl       {}       QuantLib::ActualActual::ISDA_Impl:: \rfsb       Void **         ▷ endote       _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl:: \rfsb       Void **         ▷ endote       [default]       boost::       boost::  |   |
| tos Locals Threads Modules Watch1   | tos <mark>Locals</mark> Threads Modules Watch1   |   | impl_       termStructureDayCounter       {impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         ✓        impl_       shared_ptr {} [2 strong refs] [default] }       boost::         ✓        [ptr]       0x0082ff88 {}       Quantl         ✓        [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷        QuantLib::DayCounter::Impl       {}       Quantl         ▷        _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::\ftabl       viet   |   |
| tos Locals Threads Modules Watch1   | tos Locals Threads Modules Watch1  |   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         ✓        impl_shared_ptr {} [2 strong refs] [default] }       Quantl         ✓        [ptr]       shared_ptr {} [2 strong refs] [default]       boost:         ✓        [ptr]       0x0082ff88 {}       Quantl         ✓        [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷        QuantLib::DayCounter::Impl       {}       Quantl         ▷        _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::vftab       void **   |   |
| tos Locals Threads Modules Watch1   | tos Locals Threads Modules Watch1  | 0,00104 - 0,000000000000000000000000000000000   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         I  |   |
| tos Locals Threads Modules Watch 1  | tos Locals Threads Modules Watch1  | 0.00106-70 (m. 0.0000600 ( ) m. (m. 0.00001170 d-f-ult ) )  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         I  |   |
| Locals Threads Modules Watch 1  | tos Locals Threads Modules Watch 1   | N <→ [D_1] (G_1) (G_2) (G_  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl  |   |
| Istack  | tos Locals Threads Modules Watch1  | b   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Doots:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         I  |   |
| tos Locals Threads Modules Watch1   | tos Locals Threads Modules Watch1  |   | Impl_       fimpl_=shared_ptr {} [2 strong refs] [default] }       Quantl         ✓ @ impl_       shared_ptr {} [2 strong refs] [default] }       boost:         ✓ @ [ptr]       0x0082ff88 {}       Quantl         ✓ @ [quantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷ @ QuantLib::DayCounter::Impl       {}       Quantl         ▷ @ yfotr       0x00000hd34 (Bonds-vc110-mt-pd evelconst Quantl ib::ActualActual::ISDA_Impl: \ftable vicit *:   |   |
| V     [default     boost       North     0.00195-70 (av. 0.0020599 ( ) ac. (ai. 0.00201179 default ))     boost       I Stack     V     0.00195-70 (av. 0.0020599 ( ) ac. (ai. 0.00201179 default ))  | V = [defecter and allocator]     default     BooSt       N = [Deputies]     0.00195-70 (au. 0.0092669 (c.) are (ai. 0.00921178 default))     BooSt       Nos     Locals     Threads Modules Watch1   | V = [uester and allocator] $uestallit Doost: Dooster (a) = 0.00001170 (a) (b) (b) (b) (b) (b) (b) (b) (b) (b) (b$   | Impl_ termStructureDayCounter       {impl_=shared_ptr {} [2 strong refs] [default] }       Quantl         ✓        impl_ shared_ptr {} [2 strong refs] [default] }       boost:         ✓        [ptr]       0x0082ff88 {}       Quantl         ✓        [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▶        QuantLib::DayCounter::Impl       {}       Quantl         ▶        QuantLib::DayCounter::Impl       {}       Quantl   |   |
| P     [deleter and allocator]     default     boost       Locals     [Dawling]     0x00196-70 (avx 0x0092469 (c) are (ai) 0x00921178 default))     boost       I Stack     I Stack     I Stack     I Stack  | P      [deleter and allocator]     default     boost       tos     Locals     Threads     Modules     Watch 1  | ▷ ♥ [deleter and allocator]     default     boost::   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         ✓        impl_shared_ptr {} [2 strong refs] [default] }       Quantl         ✓        [ptr]       shared_ptr {} [2 strong refs] [default] }       boost:         ✓        [ptr]       0x0082ff88 {}       Quantl         ✓        [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷        QuantLib::DayCounter::Impl       {}       Quantl  |   |
| b @ [deleter and allocator]     default     boost       b @ [Deux/Gaux]     0.00126-70 (au. 0.0082688 (c.) are (ai. 0.00821178 default))     boost       cost     Threads     Modules     Watch 1   | b @ [deleter and allocator]     default     boost       cos Locals     Threads Modules Watch 1     0x00126-70 (mm 0x0022699 (m) and (mm 0x0022699 (mm 0x0022699 (mm 0x0022699 (mm 0x0022699 (mm 0x0022699 (mm 0x0022699 (mm 0x002269 (mm 0x0022699 (mm 0x022699 (mm 0x0022699 (mm 0x0022699 (mm 0x0022699 (mm 0x0022699 (mm  | ▷ ● [deleter and allocator]     default     boost::   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} {       Quantl  |   |
| b @ [deleter and allocator]     default     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.00221172 default))     boost       b @ [deleter and allocator]     0.00126-70 (au. 0.0022622 (c.) are (ai. 0.002262 (c.) are (ai. 0.002  | b @ [deleter and allocator]     default     boost       os Locals     Threads Modules Watch 1     0.00126-70 (m. 0.0022698 ( ) m. (m. 0.0022698 ( ) m. (m. 0.00221178 default ) )  | > • [deleter and allocator]     default     boost::     boost::   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} {       Quantl  |   |
| b @ [deleter and allocator]     default     boost       b @ [Deux/Gourd]     0-00195-70 (au. 0-0082692 (c.) ac. (ai. 0-00821178 default))     boost       b @ [Deux/Gourd]     0-00195-70 (au. 0-0082692 (c.) ac. (ai. 0-00821178 default))     boost       b @ [Stack     •     •  | b @ [deleter and allocator]     default     boost       os Locals     Threads Modules Watch 1     0.00126-70 (nr. 0.0022692 (c.) nr. (nr. 0.00221172 default 1)  | ▷ ● [deleter and allocator]     default     boost::       ▷ ● [Den V[and]     0.00236789 ( ) and (all control of the second of the  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} {       Quantl  |   |
| <ul> <li> </li> <li> <li> </li> <li> </li> <li> <li></li></li></li></ul>  | ▷       [deleter and allocator]     default     boost       os     Locals     Threads     Modules     Watch 1  | ▷ ● [deleter and allocator]     default     boost::   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} {       Quantl  |   |
| Image: Second Second Control (Second Contro) (Second Control (Second Control (Second Control (Second Control  | > • •  | > • • [default     0.0000000+ [conds+(c  | impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     Doots:       Impl_shared_ptr {} [2 strong refs] [default] }     Quantl  |   |
| > • • _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void ?         > • • [deleter and allocator]       default         > • • Icals       0x00196-70 (cm. 0x0082468 (c.) cm. (cri. 0x00821178 default.))         Stack       • • • • • • • • • • • • • • • • • • •  | <ul> <li></li></ul>  | > • •vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``       > • • [deleter and allocator]     default       > • • [Den View]     0x0026720 (mn 0x00201720 default)  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         I  |   |
| • •vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: vftabl void 7         • • [deleter and allocator]       default         • os       Locals         Threads       Modules         Watch 1       Stack  | <ul> <li>vfptr</li> <li>0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: vftabl void *</li> <li>default</li> <li>default</li> <li>boost</li> </ul> os Locals         Threads         Modules         Watch 1                value              value              value              value  | > • • _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * `         > • [deleter and allocator]       default         > • • [deleter and allocator]       0.0020100 (c.) and (c.) boost:   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Doots:         Impl_shared_ptr {} [2 strong refs] [default]       Doots:         Impl_shared_ptr {} [   |   |
| V     QuantLib::DayCounter::impl     {}     QuantLib::DayCounter::impl     QuantLib::DayCounter::impl       V   | v < QuantLib::DayCounter:::mpi   | V     QuantLib::DayCounter::Impl     {}     QuantLib::DayCounter::Impl     QuantLib::DayCounter::Impl       V   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Doutst         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Doutst         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Doutst         I  |   |
| > @ QuantLib::DayCounter::Impl     {}     Quant       > @ _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void ?       > @ [deleter and allocator]     default     boost       > @ [deleter and allocator]     0x00196-70 (cm: 0x00924699 (c)) cm: (ci: 0x00921179 default ))     boost       Stack  | >      QuantLib::DayCounter::Impl     {}     Quant       >   | <ul> <li></li></ul>   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Q  |   |
| > • • QuantLib::DayCounter::Impl     {}     QuantLib::DayCounter::Impl     QuantLib::QuantLib::ActualActual::ISDA_Impl::\ftabl void '       > • • _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\ftabl void '     boost       > • • [deleter and allocator]     default     boost       os     Locals     Threads     Modules       Stack     •     •   | <ul> <li>QuantLib::DayCounter::Impl</li> <li>Quant</li> </ul> <ul> <li>QuantLib::DayCounter::Impl</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>Void*</li> </ul> <ul> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>Void*</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>Void*</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>Void*</li> <li>Coll*6-70 (m)</li> <li>Void*</li> <li>Coll*6-70 (m)</li> <li>Void*</li> <li>Coll*6-70 (m)</li> <li>Void*</li> <li>Coll*6-70 (m)</li> <li>Void*</li> </ul> <ul> <li>Void*</li> <li>Void*</li> <li>Coll*6-70 (m)</li> <li>Col*6-70 (m)</li> <licol*6-70 (m)<="" li=""> <li>Col*6-70 (m)<!--</td--><td>&gt; • • QuantLib::DayCounter::Impl       {}       QuantLib::ActualActual::ISDA_Impl::`vftabl       QuantLib::ActualActual::ISDA_Impl::`vftabl       void * :         &gt; • • [deleter and allocator]       default       boost:       boost:</td><td>Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_mode       0x0082ff88 {}</td></li></licol*6-70></ul>   | > • • QuantLib::DayCounter::Impl       {}       QuantLib::ActualActual::ISDA_Impl::`vftabl       QuantLib::ActualActual::ISDA_Impl::`vftabl       void * :         > • • [deleter and allocator]       default       boost:       boost:  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_mode       0x0082ff88 {}  |   |
| > • QuantLib::DayCounter::Impl     {}     QuantLib::DayCounter::Impl     QuantLib::QuantLib::ActualActual::ISDA_Impl::\ftabl void '       > • • _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\ftabl void '     boost       > • • [deleter and allocator]     default     boost       > tocals     Threads Modules Watch1     0x00126-70 (==: 0x00226282 (_) == (=: 0x00221178 default ))     boost       Stack     •     •   | <ul> <li>QuantLib::DayCounter::Impl             </li> <li>Quant</li> <li>Yetr</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>Void *</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>Void *</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>Void *</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>Void *</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl</li> <li>QuantLib::ActualActualActual::ISDA_Impl::Vftabl</li> <li< td=""><td>▷ ● QuantLib::DayCounter::Impl       {}       QuantLib::ActualActual::ISDA_Impl::`vftabl       QuantLib::ActualActual::ISDA_Impl::`vftabl       void * :         ▷ ● [deleter and allocator]       default       boost:       0.0005/600.(</td><td>Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:</td></li<></ul>   | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib::ActualActual::ISDA_Impl::`vftabl       QuantLib::ActualActual::ISDA_Impl::`vftabl       void * :         ▷ ● [deleter and allocator]       default       boost:       0.0005/600.(  | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:  |   |
| > • QuantLib::DayCounter::Impl     {}     QuantLib::DayCounter::Impl     QuantLib::QuantLib::ActualActual::ISDA_Impl:: \ftabl       > • • _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \ftabl     Void 7       > • • [deleter and allocator]     default     boost       os     Locals     Threads     Modules       Stack     •     •  | <ul> <li>QuantLib::DayCounter::Impl             <li>{}</li> <li>QuantLib::DayCounter::Impl             <li>(}</li> </li></li></ul> QuantLib::DayCounter::Impl <ul> <li>(</li></ul>   | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib         ▷ ● _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\vftabl void * ``         ▷ ● [deleter and allocator]       default       boost:         ▷ ● [Denu (Genu)       0x0026/070 (cm. 0x0026/070 (   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:  |   |
| > • QuantLib::DayCounter::Impl     {}     QuantLib::DayCounter::Impl     QuantLib::QuantLib::ActualActual::ISDA_Impl:: \ftabl       > • • _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \ftabl     Void 7       > • • [deleter and allocator]     default     boost       os     Locals     Threads     Modules       Stack     •     •  | <ul> <li>QuantLib::DayCounter::Impl             <li>{}</li> <li>QuantLib::DayCounter::Impl             <li>(}</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>(default</li> <li>(afault</li> <li>(DoubleGault (Counter))</li> <li>(Doub</li></li></li></ul>  | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib         ▷ ● _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\vftabl void * ``         ▷ ● [deleter and allocator]       default       boost:         ▷ ● [Denu (Genu)       0x0026/070 (cm. 0x0026/070 (   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_int     shared_ptr {} [2 strong refs] [default]   |   |
| > • QuantLib::DayCounter::Impl     {}     QuantLib::DayCounter::Impl     QuantLib::QuantLib::ActualActual::ISDA_Impl:: \ftabl       > • • _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \ftabl     Void 7       > • • [deleter and allocator]     default     boost       os     Locals     Threads     Modules       Stack     •     •  | <ul> <li>QuantLib::DayCounter::Impl             <li>{}</li> <li>QuantLib::DayCounter::Impl             <li>(}</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>(default</li> <li>(afault</li> <li>(DoubleGault (Counter))</li> <li>(Doub</li></li></li></ul>  | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib         ▷ ● _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\vftabl void * ``         ▷ ● [deleter and allocator]       default       boost:         ▷ ● [Denu (Genu)       0x0026/070 (cm. 0x0026/070 (   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_int     shared_ptr {} [2 strong refs] [default]   |   |
| > • QuantLib::DayCounter::Impl     {}     QuantLib::DayCounter::Impl     QuantLib::QuantLib::ActualActual::ISDA_Impl:: \ftabl       > • • _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \ftabl     Void 7       > • • [deleter and allocator]     default     boost       os     Locals     Threads     Modules       Stack     •     •  | <ul> <li>QuantLib::DayCounter::Impl             <li>{}</li> <li>QuantLib::DayCounter::Impl             <li>(}</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>(default</li> <li>(afault</li> <li>(DoubleGault (Counter))</li> <li>(Doub</li></li></li></ul>  | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib         ▷ ● _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``         ▷ ● [deleter and allocator]       default       boost:         ▷ ● [Denu (Genu)       0x0026/070 (cm. 0x0026/070 (   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:  |   |
| > • QuantLib::DayCounter::Impl     {}     Quant       > •vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \ftabl void '       > • • [deleter and allocator]     default       > b • [deleter and allocator]     0x00186=70 (cm. 0x00826688 (c.) cm. (cm. 0x00821178 default ))       > b • Instant     0x00186=70 (cm. 0x00826688 (c.) cm. (cm. 0x00821178 default ))  | <ul> <li>QuantLib::DayCounter::Impl             <li>{}</li> <li>QuantLib::DayCounter::Impl             <li>(}</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>(default</li> <li>(fault for the default)</li> <li>(fault for the default for the defa</li></li></li></ul>  | ▷ @ QuantLib::DayCounter::Impl       {}       QuantLib         ▷ @ _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\vftabl void * ``         ▷ @ [deleter and allocator]       default       boost:         ▷ @ [Deru Koral       0x0026786 (L) and (L)   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:  |   |
| <ul> <li>QuantLib::DayCounter::Impl</li> <li>{}</li> <li>QuantLib::DayCounter::Impl</li> <li>{}</li> <li>QuantLib::DayCounter::Impl</li> <li>{}</li> <li>QuantLib::DayCounter::Impl</li> <li>{}</li> <li>QuantLib::DayCounter::Impl</li> <li>QuantLib::DayCounte</li></ul>  | QuantLib::DayCounter::Impl     {}     Quant     QuantLib::DayCounter::Impl     {}     Quant     Quant     Quant     QuantLib::ActualActual::ISDA_Impl::Vftabl     Void *     default     default     boost     Locals     Threads     Modules     Watch 1  | ▷ @ QuantLib::DayCounter::Impl       {}       Quantl         ▷ @ _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``         ▷ @ [deleter and allocator]       default       boost:         ▷ @ [Deru Koral       0x0026789 () concentration ()       0x0026789 () concentration ()   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:  |   |
| > • QuantLib::DayCounter::Impl     {}     Quant       > • •vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void 7     > • • [deleter and allocator]     default       > • • [deleter and allocator]     default     > 0x00126:20 (cm: 0x0022628 ( ) cm: (cm: 0x00221178 default ))     > • • • • • • • • • • • • • • • • • •  | <ul> <li>QuantLib::DayCounter::Impl</li> <li>Quant</li> </ul> <ul> <li>QuantLib::DayCounter::Impl</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>Quant</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>Quant</li> <li>Quant</li> <li>QuantLib::ActualActual::ISDA_Impl::Vftabl void *</li> <li>Quant</li> <li>Quant</li></ul>   | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib::DayCounter::Impl       {}         ▷ ● _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``       b         ▷ ● [deleter and allocator]       default       boost:       boost:   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:  |   |
| > • QuantLib::DayCounter::Impl     {}     Quant       > • •vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \ftabl void '       > • • [deleter and allocator]     default       > • • [delexer and allocator]     0x00186=70 (cm: 0x0082668 (c.) cm: (ci: 0x00821178 default ))       os     Locals       Threads     Modules       Vare     Vare  | <ul> <li>QuantLib::DayCounter::Impl             <li>{}</li> <li>QuantLib::DayCounter::Impl             <li>(}</li> </li></li></ul> QuantLib::DayCounter::Impl <ul> <li>(</li></ul>   | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib         ▷ ● _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``         ▷ ● [deleter and allocator]       default         ▷ ● [March Gard       0x002000000000000000000000000000000000   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:  |   |
| > • QuantLib::DayCounter::Impl     {}     Quant       > • •vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \ftabl void '       > • • [deleter and allocator]     default       > • • [deleter and allocator]     0x00126=70 (and 0x0026682 (all and (all control))       > • • Coals     Threads       Modules     Watch 1  | \u03c9         QuantLib::DayCounter::Impl         {}         Quant         \u03c9         _vfptr         0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\v1tabl void *   | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib         ▷ ● _vfptr       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``         ▷ ● [deleter and allocator]       default         ▷ ● [March Card       0x0060bd34 (Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default]       Boost:         Impl_shared_ptr {} [2 strong ref   |   |
| QuantLib::DayCounter::Impl     {}     Quan     QuantLib::DayCounter::Impl     {}     Quan     QuantLib::DayCounter::Impl     {}     Quan     QuantLib::ActualActual::ISDA_Impl::Vftabl     void 7     P @ [deleter and allocator]     default     Do0195-70 (cm: 0x00924599 (c.) cm: (ci: 0x00921179 default ))     Stack     Name  | QuantLib::DayCounter::Impl     {}     Quant     QuantLib::DayCounter::Impl     {}     Quant  | ▷ ● QuantLib::DayCounter::Impl       {}       QuantLib::DayCounter::Impl       {}         ▷ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void * ``       b         ▷ ● [deleter and allocator]       default       boost:       boost:   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:       Impl_shared_ptr {} [2 strong refs] [default]     boost:  |   |
| ↓      QuantLib::DayCounter::Impl     {}     QuantLib::DayCounter::Impl     QuantLib::ActualActual::ISDA_Impl:: \ftabl       ↓  | ▷ ○ QuantLib::DayCounter::Impl       {}       Quant         ▷ ○ _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftabl void *       Quant         ▷ ○ [deleter and allocator]       default       boost         □ coals       Threads       Modules       Watch 1  | ▷ ○ QuantLib::DayCounter::Impl       {}       QuantLib::DayCounter::Impl       {}         ▷ ○ _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\vftabl void * ``       QuantLib::QuantLib::ActualActual::ISDA_Impl::\vftabl void * ``         ▷ ○ [deleter and allocator]       default       0x0026786 () constant       boost:   | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost::         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Doost::         Impl_shared_ptr {} [2 strong refs] [default] }       Doost:: <t< td=""></t<>  |   |
| Image: Control of Contro of Control of Control of Control of Control of Control of   | -     [QuantLib::ActualActual::ISDA_Impi]     {}     Quant       ▶     ●     QuantLib::DayCounter::Impi]     {}     Quant       ▶     ●     _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl:: \ftabl void *       ▶     ●     [deleter and allocator]     default     boost       bost     Locals     Threads     Modules     Watch 1   | Image: Construction actual  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost:         Impl_shared_ptr {} [2 strong refs] [default] }       Duantl  |   |
|   | ▲ ● [QuantLib::ActualActual::ISDA_Impl]     {}     Quant       ▶ ● QuantLib::DayCounter::Impl     {}     Quant       ▶ ● _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void *     Void *       ▶ ● [deleter and allocator]     default     boost       ▶ ● Locals     Threads     Modules     Watch 1  | ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       QuantLib::DayCounter::Impl       QuantLib::DayCounter::Impl       QuantLib::DayCounter::Impl       QuantLib::OayCounter::Impl       QuantLib::OayCounter::Impl::Vitabl       Void * 1         ▷ ●   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     Doct       Impl_shared_ptr {} [2 strong refs] [default]     Doct  |   |
|   | ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▶ ● QuantLib::DayCounter::Impl       {}       Quant         ▶ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl void *       Void *         ▶ ● [deleter and allocator]       default       boost         ▶ ● [locals       Threads Modules Watch 1       0x00126-70 (mu 0x0025698 (L) ma (mi 0x00231178 default))       boost  | ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       QuantLib::DayCounter::Impl       QuantLib::DayCounter::Impl       QuantLib::QuantLib::ActualActual::ISDA_Impl::`vftabl       QuantLib::ActualActual::ISDA_Impl::`vftabl       QuantLib::ActualActual::ISDA_Impl::`vftabl       Void * ``         ▷ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::`vftabl       Void * ``         ▷ ● [deleter and allocator]       default       boost::       boost::   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl       Impl_shared_ptr {} [2 strong refs] [default] }     boost:  |   |
| ▲ ● [QuantLib::ActualActual::ISDA_Impl]     {}     QuantLib::DayCounter::Impl     QuantLib::DayCounter::Impl     QuantLib::ActualActual::ISDA_Impl:: \ftable     Void '       ▶ ● [deleter and allocator]     default     0:00196:70 (m:: 0:0092)ff88 () == (m:: 0:0092)1178 defoult )     boost       os     Locals     Threads     Modules     Watch 1  | ✓ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▷ ● QuantLib::DayCounter::Impl       {}       Quant         ▷ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftabl void *       Quant         ▷ ● [deleter and allocator]       default       boost         ▷ ● [deleter and allocator]       0x00186=70 (m. 0x00826688 (_) mm (m. 0x0082688 (_)   | ✓        [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷        QuantLib::DayCounter::Impl       {}       Quantl         ▷  | Impl_shared_ptr {} [2 strong refs] [default] }       Quantl         Impl_shared_ptr {} [2 strong refs] [default] }       boost::   |   |
| ▲ ● [ptr]     UXU02/T88 {}     Quan       ▲ ● [QuantLib::ActualActual::ISDA_Impl]     {}     Quan       ▶ ● QuantLib::DayCounter::Impl     {}     Quan       ▶ ● _vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void *     Quan       ▶ ● [deleter and allocator]     default     boost       ▶ ● [deleter and allocator]     0x00186-70 (== 0x0082468 (_) == (=i = 0x00821178 default ))     boost       Stack     •   | ▲ ♥ [ptr]     0x0082tt88 {}}     Quant       ▲ ♥ [QuantLib::ActualActual::ISDA_Impl]     {}     Quant       ▶ ♥ QuantLib::DayCounter::Impl     {}     Quant       ▶ ♥  | ▲ ♥ [ptr]     UXUU82tt88 {}     Quantl       ▲ ♥ [QuantLib::ActualActual::ISDA_Impl]     {}     Quantl       ▶ ♥ QuantLib::DayCounter::Impl     {}     Quantl       ▶ ♥vfptr     0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::'vftabl void * '       ▶ ♥ [deleter and allocator]     default       ▶ ♥ [deleter and allocator]     0x0060bd34 {Complexelocation (c) = 0x000000000 (c) = 0x000000000 (c) = 0x000000000 (c) = 0x000000000 (c) = 0x0000000000 (c) = 0x0000000000 (c) = 0x0000000000 (c) = 0x0000000000 (c) = 0x0000000000000000000000000000000000  | termStructureDayCounter     {impl_=shared_ptr {} [2 strong refs] [default] }     Quantl     shared_ptr {} [2 strong refs] [default] }  |   |
| ▲ ● [ptr]       0x0082ff88 {}       Quan         ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quan         ▶ ● QuantLib::DayCounter::Impl       {}       Quan         ▶ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void 7       Quan         ▶ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void 7       boost         ▶ ● [deleter and allocator]       default       boost       boost         Locals       Threads       Modules       Watch 1   | ▲ ● [ptr]       0x0082ff88 {}       Quant         ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▶ ● QuantLib::DayCounter::Impl       {}       Quant         ▶ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftabl void *         ▶ ● [deleter and allocator]       default       boost         os Locals       Threads Modules Watch 1       0x0082ff88 {}       bconstruction   | ▲ ● [ptr]       0x0082ff88 {}       Quantl         ▲ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▶ ● QuantLib::DayCounter::Impl       {}       Quantl         ▶ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\ftabl void * *         ▶ ● [deleter and allocator]       default       boost:         ▶ ● [Day Mond       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::\ftabl void * *   | Impl_shared_ptr {} [2 strong refs] [default] }     Quantl  |   |
| Image       Image (in ) (2 strong rens) (default)       Doos         Image (ptr)       0x0082ff88 {}       Quan         Image (ptr)       0x0080bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void *       Quan         Image (ptr)       0x0080bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void *       boost         Image (ptr)       0x0080bd34 {Bonds-vc10-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl::Vftabl void *       boost   | → • • • • • • • • • • • • • • • • • • •  | → ♥ [ptr]       0x0082ff88 {}       Quantl         → ♥ [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▶ ♥ QuantLib::DayCounter::Impl       {}       Quantl         ▶ ♥ QuantLib::DayCounter::Impl       {}       Quantl         ▶ ♥ QuantLib::DayCounter::Impl       {}       Quantl         ▶ ♥ [deleter and allocator]       default       boost:         ▶ ♥ [Day King]       Quantl       0.000010000000000000000000000000000000   | termStructureDayCounter     {impl_=shared_ptr {} [2 strong refs] [default] }     Quantl  |   |
| Impl_       shared_ptr {} [2 strong refs] [default]       boost         Impl_       0x0082ff88 {}       Quan         Impl_       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl:       Vftabl void ?         Impl_       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::       Vftabl void ?         Impl_       0x00186=70 (mu 0x0082ff88 {}) mpl.       boost         Impl_       0x00186=70 (mu 0x0082ff88 {}) mpl.       boost <td< td=""><td>▲ ♀ impl_       shared_ptr {} [2 strong refs] [default]       boost         ▲ ♀ [ptr]       0x0082ff88 {}       Quant         ▲ ♀ [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quant         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quant         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quant         ▷ ♀ _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftabl       void *         ▷ ♀ [deleter and allocator]       default       boost         os       Locals       Threads       Modules       Watch 1</td><td>▲ ♥ impl_       shared_ptr {} [2 strong refs] [default]       boost::         ▲ ♥ [ptr]       0x0082ff88 {}       Quantl         ▲ ♥ [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▶ ♥ QuantLib::DayCounter::Impl       {}       Quantl         ▶ ♥vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftab       void * 1         ▶ ♥ [deleter and allocator]       default       boost::       0x0021178 d (suit h)</td><td></td></td<>   | ▲ ♀ impl_       shared_ptr {} [2 strong refs] [default]       boost         ▲ ♀ [ptr]       0x0082ff88 {}       Quant         ▲ ♀ [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quant         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quant         ▶ ♀ QuantLib::DayCounter::Impl       {}       Quant         ▷ ♀ _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftabl       void *         ▷ ♀ [deleter and allocator]       default       boost         os       Locals       Threads       Modules       Watch 1   | ▲ ♥ impl_       shared_ptr {} [2 strong refs] [default]       boost::         ▲ ♥ [ptr]       0x0082ff88 {}       Quantl         ▲ ♥ [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▶ ♥ QuantLib::DayCounter::Impl       {}       Quantl         ▶ ♥vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftab       void * 1         ▶ ♥ [deleter and allocator]       default       boost::       0x0021178 d (suit h)   |  |   |
| ▲ ♀ impl_       shared_ptr {} [2 strong refs] [default]       boost         ▲ ♀ [ptr]       0x0082ff88 {}       Quant         ▲ ♀ [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▷ ♀ QuantLib::DayCounter::Impl       {}       Quant         ▷ ♀ _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \vftabl void ?       Void?         ▷ ♀ _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exe!const QuantLib::ActualActual::ISDA_Impl:: \vftabl void ?       boost         ▷ ♀ [deleter and allocator]       default       boost       boost         Locals       Threads       Modules       Watch 1         Stack   | ▲ Implchickspit (m) [2 strong refs] [default]       boost         ▲ Implchickspit (m) [2 strong refs] [default]       Quant         ▶ Implchickspit (m) [2 strong refs] [default]       Default         ▶ Implchickspit (m) [2 strong refs] [default]       Default         > Implchickspit (m) [2 strong refs] [default]  | ✓ ♀ impl_       shared_ptr {} [2 strong refs] [default]       boost::         ✓ ♀ [ptr]       0x0082ff88 {}       Quantl         ✓ ♀ [QuantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷ ♀ QuantLib::DayCounter::Impl       {}       Quantl         ▷ ♀ _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftabl       void * 1         ▷ ♀ _(deleter and allocator]       default       boost::       0c00201178 d for th b)   | NULL VOILE   |   |
| impl_shared_ptr {} [2 strong refs] [default] }       Quant         Impl_shared_ptr {} [2 strong refs] [default] }       boost         Impl_shared_ptr {} {       QuantLib::ActualActual::ISDA_Impl] {} }       QuantLib::DayCounter::Impl {} Quant         Impl_shared_ptr {} {       Impl_shared_ptr {}        0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftabl void 7         Impl_shared_ptr {} {       0x00186:70 (== 0x0082688 {} = (= 0x00821178 default ))       boost         Impl_shared_ptr {} {       0x00186:70 (== 0x0082688 {} = (= 0x00821178 default ))       boost         Impl_shared_ptr {} {       0x00186:70 (== 0x0082688 {} = (= 0x00821178 default ))       boost         Impl_shared_ptr {} {       0x00186:70 (== 0x0082688 {} = (= 0x00821178 default ))       boost         Impl_shared_ptr {} {       0x00186:70 (== 0x0082688 {} = (= 0x00821178 default ))       boost         Impl_shared_ptr {} {       0x0082688 {} = (= 0x00821178 default ))       boost         Impl_shared_ptr {} {       0x0082688 {} = (= 0x00821178 default ))       boost  | impl_shared_ptr {} [2 strong refs] [default] }       Quant         ✓ ♀ impl_       shared_ptr {} [2 strong refs] [default] }       boost         ✓ ♀ [ptr]       0x0082ff88 {}       Quant         ✓ ● [QuantLib::ActualActual::ISDA_Impl]       {}       Quant         ▷ ● QuantLib::DayCounter::Impl       {}       Quant         ▷ ● _vfptr       0x0060bd34 {Bonds-vc110-mt-gd.exelconst QuantLib::ActualActual::ISDA_Impl::Vftabl void *       boost         ▷ ● [deleter and allocator]       default       boost       boost         Locals       Threads Modules Watch 1       0x00186=70 ( 0x00204682 () == ( 0x00201178 default ))       boost   | implshared_ptr {} [2 strong refs] [default] }       Quantl         ✓ ♀ impl_       shared_ptr {} [2 strong refs] [default] }       boost::         ✓ ♀ [ptr]       0x0082ff88 {}       Quantl         ✓ ● [quantLib::ActualActual::ISDA_Impl]       {}       Quantl         ▷ ● QuantLib::DayCounter::Impl       {}       Quantl         ▷ ●  | ame Value Type   |   |

# Chapter 2: Recalling object oriented programming and getting started with date arithmetic.

Although this book assumes the basic knowledge of C++ (and thus OOP) we will at first quickly review its concepts. My experience shows that the students usually understand the main OOP principles but do not fully realize their advantages and their proper usage. The object oriented programming, as its name implies, works with *objects*. In turn, the objects are the instances of *classes*. Let us start with the class *Date* that is defined in ql/time/date.hpp

| Open File in Solution [8 of 2214]    |                | Note + 7,<br>August + 8,  | ৢ <mark>×</mark> |
|--------------------------------------|----------------|---|------------------|
| File                                 | Project        | Path  | Modified         |
| ++ date.cpp                          | QuantLib_vc11  | D:\QuantLibBOOK\QL\QuantLib-1.4.1\ql\time\date.cpp                        | 12/12/201        |
| h date.hpp                           | QuantLib_vc11  | D:\QuantLibBOOK\QL\QuantLib-1.4.1\ql\time\date.hpp                        | 12/12/201        |
| ++ dategenerationrule.cpp            | QuantLib_vc11  | D:\QuantLibBOOK\QL\QuantLib-1.4.1\ql\time\dategenerationrule.cpp          | 12/12/201        |
| h dategenerationrule.hpp             | QuantLib_vc11  | D:\QuantLibBOOK\QL\QuantLib-1.4.1\ql\time\dategenerationrule.hpp          | 12/12/201        |
| ++ dateinterval.cpp                  | QuantLib_vc11  | D:\QuantLibBOOK\QL\QuantLib-1.4.1\ql\experimental\commodities\dateinte    | 12/12/201        |
| h dateinterval.hpp                   | QuantLib_vc11  | D:\QuantLibBOOK\QL\QuantLib-1.4.1\ql\experimental\commodities\dateinte    | 12/12/201        |
| ++ dates.cpp                         | testsuite_vc11 | D:\QuantLibBOOK\QL\QuantLib-1.4.1\test-suite\dates.cpp                    | 12/12/201        |
| h dates.hpp                          | testsuite_vc11 | $\label{eq:def-D} D:\QuantLibBOOK\QL\QuantLib-1.4.1\test-suite\dates.hpp$ | 12/12/201        |
|                                      |                |   |                  |
| date                                 |                |   | ОК               |
| Show only files in the current solut | ion            |   | Cancel           |
| gure 1.1                             |                |   | r VS Tip®        |

In order to open date.hpp quickly go to VASSISTX -> Open File in Solution and type date

As every class, Date contains <u>fields</u> (a.k.a. attributes or member variables) and <u>methods</u> (the procedures and functions defined within the class). There is only one field: *serialNumber\_* that contains the number of days counting from 01.12.1900. The understrike at the end of the field's name is a coding convention, which lets us quickly distinguish the class member variables from the local variables in class methods<sup>22</sup>.

In accordance with the *encapsulation* (the 1st OOP principle) serialNumber\_ is *private*, which means that only the methods of the *Date* class<sup>23</sup> can access this field. There are also some private methods, for example *static void checkSerialNumber(BigInteger serialNumber)*. This method, as its name implies, checks whether a serial number (which uniquely represents a date) is valid.

Public and private methods separate, so to say, the interface and the realization. In this sense the private methods not only say that you are not allowed to call them but first of all they tell you that you will <u>never</u> <u>need</u> to call them! Indeed, you will not need *checkSerialNumber(BigInteger serialNumber)* when you are calculating any financial instrument. But obviously you may need the methods *dayOfMonth(), month(), year()* that are declared public. Moreover, QuantLib core team can change the implementation of *checkSerialNumber(BigInteger serialNumber)* or even rename/replace this method without a fear that it may damage the software projects that engage QuantLib.

<sup>&</sup>lt;sup>22</sup> Probably a better convention would be an understike at the beginning, i.e. "\_serialNumber" instead of "serialNumber\_". Then the Visual Studio Debugger would group all member variables together. However, you should better follow the established convention, especially if you are going to contribute your code to QuantLib.

<sup>&</sup>lt;sup>23</sup> Additionally, the [methods of] the *friend* classes can. But *Date* has no friend classes.

You may have also noted that some methods are declared as *static* and some not. The latter have to do with objects whereas the former are defined for the classes as such. For example, a method call like *Date.weekday()* does not make sense since *Date* is a class (and not an object of the class *Date*). Thus trying to call *Date.weekday()* one actually says "tell me the weekday of date" (which date?!). But the command "tell me the weekday of 10.12.1979"<sup>24</sup> is absolutely correct.

On the other hand some properties of *Date* are the same for *all* objects of the class Date. Thus we can deal with them at the class (not at the object) level. For example the methods minDate() and maxDate() return the minimal and maximal dates, which QuantLib can digest (they are, respectively, 01.01.1901 and 31.12.2199). These are obviously class-level attributes. On the other hand the method isEndOfMonth(const Date& d) is also declared as static but it is not obvious whether it really belongs to the class level. One may also have defined it as non-static, since it is the property of a concrete date, whether it is the end of month or not. Moreover, I had an occasion to maintain a proprietary library that calculated loans and the date class in this library had an attribute endOfMonth . The problem was as following: if a credit starts, say on the 28th of February and the installments are paid monthly, when should the next installment be paid: on the 28th or on the 31th of March?! It actually depended on loan agreement (ultimo or not). At first glance the introduction of the attribute *endOfMonth* in the *Date* class seems ugly and a better solution would be to introduce something like *ultimo\_* attribute in the class *Loan*. However, there were so many types of loans and their hierarchy was so complicated that the solution with the endOfMonth\_ as Date attribute was probably the easiest from the practical point of view. In other words: a good software architecture is very important but in practice there are always trade-offs between architecture, performance, code brevity and (last but not list) programmers' skills and personal preferences.

So far let us stop with theory for a while and do something with QuantLib. First of all add a new project *DateToy* to the QuantLib Solution. In Solution Explorer click on Solution 'QuantLib\_vc11' -> Add -> New Project.

| Add New Project  | hand genetic-Ch                             | S X                           |
|--|---|-------------------------------|
| ▷ Recent   | .NET Framework 4.5    Sort by: Default      | 🔹 🏥 🔚 Search Installed Te 🔎 🗸 |
| Installed  | Win32 Console Application Visual C++        | Type: Visual C++              |
| ATL<br>CLR   | MFC Application Visual C++                  | application                   |
| General<br>MFC   | Win32 Project Visual C++                    |                               |
| Test<br>Win32  | Empty Project Visual C++                    |                               |
| <ul> <li>Other Languages</li> <li>Other Project Types</li> </ul> | Makefile Project Visual C++                 |                               |
| ▶ Online   |   |                               |
|  | Click here to go online and find templates. |                               |
| Name: DateToy  |   |                               |
| Location:  | OK\QL\QuantLib-1.4.1\Examples\              | Browse OK Cancel              |
|  |   |                               |
| Figure 1.2 Adding new project                                    | to the QuantLib Solution                    |                               |

<sup>&</sup>lt;sup>24</sup> In terms of C++/QuantLib: (Date(10, December, 1979)).weekday();

*QuantLib::Date(10, QuantLib::December, 1979)* instantiates the object of the class *Date* and *weekday()* returns the weekday of 10.12.1979 (it is Monday).

Select *Console application* and deactivate *Precomiled headers* and *Security DevelopmentLifecycle (SDL) checks* (Figure 1.3).

| Win32 Application Wizard - Date  | Гоу   |                              | 8 x    |
|----------------------------------|---|------------------------------|--------|
| Application                      | on Settings   |                              |        |
| Overview<br>Application Settings | Application type:<br><ul> <li>Windows application</li> <li>Console application</li> <li>DLL</li> <li>Static library</li> </ul> Additional options: <ul> <li>Empty project</li> <li>Export symbols</li> <li>Precompiled header</li> <li>Security Development Lifecycle (SDL) checks</li> </ul> | Add common header files for: |        |
|                                  | < Previous  | Next > Finish                | Cancel |
| Figure 1.3 Application settings  | for the new project   |                              |        |

| DateToy Property Pages        |  | ? ×                                |
|-------------------------------|--|------------------------------------|
| <u>C</u> onfiguration: N/A    | → Platform:                            | N/A                                |
| Common Properties             | <u>R</u> eferences:                    |                                    |
| References                    | Name                                   | Build Properties                   |
| Configuration Properties      | Quantl ib                              | Copy Local False                   |
|                               | Quancelo                               | Copy Local Satellite Asser False 👻 |
|                               |  | Build Properties                   |
|                               | Add <u>N</u> ew Reference R <u>e</u> r | nove Reference                     |
|                               |  | OK Cancel Apply                    |
| Figure 1.4 Add reference to ( | QuantLib                               |                                    |

Click Finish, a new project DateToy will appear in Solution Explorer. Click on this project, choose Properties -> Common Properties -> References and add reference to QuantLib (Figure 1.4).

Choose Properties -> Configuration Properties -> C++ and set Additional Include Directories to ..\..;%(AdditionalIncludeDirectories) (Figure 1.5).

| DateToy Property Pages Configuration: Active(Debug)  | Platform: Active(Win  | 32)  Configuration Manager                            |
|--|---|---|
|  | Additional Include Directories<br>Additional #using Directories<br>Debug Information Format<br>Common Language RunTime Suppo<br>Consume Windows Runtime Extensic<br>Suppress Startup Banner<br>Warning Level<br>Treat Warnings As Errors<br>SDL checks<br>Multi-processor Compilation | And               |
| <ul> <li>Linker</li> <li>Manifest Tool</li> <li>XML Document Genera</li> <li>Browse Information</li> <li>Build Events</li> </ul> | Additional Include Directories<br>Specifies one or more directories to add to<br>than one. (/I[path])   | o the include path; separate with semi-colons if more |

Finally set the boost paths like we did before (Figure 0.4).

```
Figure 1.5 Adultion
```

```
#include <iostream>
#include <ql/time/date.hpp>
using namespace QuantLib;
int main()
{
       Date veryFirstDate(1, January, 1901);
       Date anotherVeryFirstDate(367);
       Date yetAnotherVeryFirstDate(veryFirstDate);
       std::cout << (Date(10, December, 1979)).weekday() << "\n";</pre>
       std::cout << veryFirstDate << "\n";</pre>
       std::cout << anotherVeryFirstDate << "\n";</pre>
       std::cout << yetAnotherVeryFirstDate << "\n";</pre>
       std::cout << Date::minDate() << "\n";</pre>
       std::cout << Date::maxDate() << "\n";</pre>
       return 0;
```

Code 1.1 Getting started with Date class

Now we call some methods of the *Date* class we discussed before. Open file *DateToy.cpp* and type Code 1.1. Set a breakpoint by "return 0;" line. Run the project. As you can see, the objects *veryFirstDate*, *anotherVeryFirstDate* and *yetAnotherVeryFirstDate* are essentially the same. But in order to instantiate them we have called three different constructors<sup>25</sup>. In the first case we specified day, month, year and in the second case we explicitly specified the serialNumber. Finally, in the third case we engaged a so-called copy constructor (which, by the way, is not implemented in *Date* and thus automatically generated by C++ compiler)<sup>26</sup>. Class *Date* also contains the definition of the *operator<<* or, more exactly, *std::ostream& operator<<(std::ostream& out, const Date& d)*. It allows to print the date on console by means of *std::cout <<* 

Note that we write "::" to call the static methods.

Now let us add *veryFirstDate*, *anotherVeryFirstDate* and *yetAnotherVeryFirstDate* to the watch window. During debugging we can see (and even change) the properties of the watched objects. However, there is an annoying problem: since the dates are defined in terms of their serial numbers, we cannot really see much. Well, for the object *veryFirstDate* we have manually added the watches *veryFirstDate.year(),veryFirstDate.month()* and *veryFirstDate.dayOfMonth()* and thus we can see something. However, it is very cumbersome and each time we need manually refresh this watches since the Visual Studio does not update them automatically<sup>27</sup>.

| atch 1   |                      |                 |  |  |  |  |  |
|--|----------------------|-----------------|--|--|--|--|--|
| Name   | Value                | Туре            |  |  |  |  |  |
| 👂 🧉 anotherVeryFirstDate                       | {serialNumber_=367 } | QuantLib::Date  |  |  |  |  |  |
| 🔺 🤗 veryFirstDate                              | {serialNumber_=367 } | QuantLib::Date  |  |  |  |  |  |
| GerialNumber_                                  | 367                  | long            |  |  |  |  |  |
| vetAnotherVeryFirstDate                        | {serialNumber_=367 } | QuantLib::Date  |  |  |  |  |  |
| veryFirstDate.year()                           | 1901                 | int int         |  |  |  |  |  |
| veryFirstDate.month()                          | January (1)          | QuantLib::Month |  |  |  |  |  |
| veryFirstDate.dayOfMonth()                     | 1                    | int             |  |  |  |  |  |
| Autos Locals Threads Modules Watch1            |                      |                 |  |  |  |  |  |
| igure 1.6 Inspecting Date objects by debugging |                      |                 |  |  |  |  |  |

In introduction we have mentioned natvis but in this case it would not help because its ability to call methods or run its own macros is very limited. But even if it would, we were still unable to flexibly *alter* the values of the member variables. In Appendix A we hack QuantLib and refactor the implementation of *Date*. This case is an excellent example of a *practical* refactoring: we are going to improve the program with little effort (relying on already existing business logic), have a nice idea how to do it ... and confront with the fact, that our idea is not completely chaste and requires some modification of the current business logic. However, if you are more interested in financial calculations than in programming aspects, you may skip Appendix A.

<sup>&</sup>lt;sup>25</sup> Constructor is a special class method, which instantiates an object. Typically, a constructor validates input parameters (e.g. in our case day, month, year) and respectively assigns the class fields. A class can have many constructors (with different arguments).

<sup>&</sup>lt;sup>26</sup> It is a very good practice to implement the copy constructor and assignment operator in your classes. If you don't C++ compiler will generate the default implementation (a bitwise copy of member variables). It may cause problems if e.g. your class contains pointers.

<sup>&</sup>lt;sup>27</sup> Though it actually could since all these methods are *const*, i.e. they cannot change any class properties (except those that are declared as *mutable*).

Now let us finally do some financial calculations. Probably one of the most frequent (and simplest) is the calculation of a zero bond yield. However, even this simple calculation is not so easy. At first let us do the calculation and then analyze the code in detail.

```
#include <iostream>
#include <ql/quantlib.hpp>
#include <math.h>
using namespace QuantLib;
void main()
{
       //create a zero coupon bond
       Date issueDate(2, October, 2014);
       Natural settlementDays = 2;
       Calendar calendar = Germany(Germany::Settlement);
       BusinessDayConvention paymentConvention = Following;
       Real faceAmount = 100;
       Real redemption = 100;
       Date maturityDate(2, October, 2016);
       ZeroCouponBond zeroBond(settlementDays, calendar, faceAmount, maturityDate,
              paymentConvention, redemption, issueDate);
       //calculate its yield
       Real cleanPrice = 97.5;
       DayCounter dayCounter = ActualActual(ActualActual::ISDA);
       Compounding compounding = Compounded;
       Frequency frequency = Annual;
       Real accuracy = 1.0e-15;
       Size maxEvaluations = 10000;
       Rate yield = zeroBond.yield(cleanPrice, dayCounter, compounding,
              frequency, zeroBond.settlementDate(issueDate), accuracy, maxEvaluations);
       std::cout << "yield from QuantLib</pre>
                                           " << (double)yield << "\n";</pre>
       //and check the results manually
       double yearFraction2014 = (Date(31, December, 2014) - Date(7, October, 2014) + 1);
       yearFraction2014 /= 365;
       double yearFraction2016 = (Date(4, October, 2016) - Date(1, January, 2016));
       yearFraction2016 /= 366;
       double t = yearFraction2014 + 1.0 + yearFraction2016;
       double tDC = dayCounter.yearFraction(Date(7, October, 2014), Date(4, October, 2016));
       assert(abs(t - tDC) < accuracy);</pre>
       double yield2 = exp((Log(redemption) - Log(cleanPrice)) / t) - 1;
       std::cout << "yield from manual check " << yield2 << "\n";</pre>
       assert(abs((double)yield - yield2) < accuracy);</pre>
}
Code 1.2 Calculating yield of a zero bond
```

As you can see, in order to define a zero bond we need the following parameters:

1) *issueDate* is the date on which the bond is issued.

2) *settlementDays* are days that the backoffice needs to commit the transaction<sup>28</sup>. In other words, if an investor buys a bond on issueDate, he will get it on (issueDate + settlementDays). In Germany two settlement days are common for the bonds. This rule is also known as T+2. Note that the interest starts accruing from (issueDate + settlementDays), not from the issueDate! The smallest time span for which the interest is accrued is one day.

Note that the variable "settlementDays" has the type "Natural". In Visual Studio, click on "Natural" and press Alt+F12 (Peek Definition). You will see in types.hpp the following definition:

typedef unsigned QL\_INTEGER Natural; In turn, QL\_INTEGER is nothing else but int. For QuantLib it is common to actively (ab)use the definition of new types. At first it really irritates but gradually a developer gets use to it and then it even becomes helpful.

3) calendar and paymentConvention. Normally, the interest accrues both for business days and holidays. But what if a settlement or redemption falls on a holiday (and how we detect it)?! The holidays are defined by calendar; ql/time/calendars contains the calendars for many countries. In our example we considered the basic German calendar (Germany::Settlement). Besides, there are special calendars for the most important German exchanges: FrankfurtStockExchange, Xetra, Eurex, Euwax (have a look at germany.hpp). The paymentConvention tells us what to do if a payment (or settlement) falls on a holiday. In our case the convention is "Following", which means that the date will be moved to the nearest following business day. In our example the issueDate (2.10.2014) is Thursday. Thus T+2, i.e. the settlement date (4.10.2014) falls on

Saturday, which is not a business day. We need two business date for the settlement, so the settlement is, according to the "Following" rule is 7.10.2014 (Tuesday).

Moreover, the maturity date (2.10.2016) is also a non-business day (Sunday). According to the "Following" rule the bond should be redeemed on 3.10.2014 ... but it is a holiday according to our calendar<sup>29</sup> thus the maturity date is shifted to 4.10.2014.

4) *faceAmount* and *redemption* are self-explanatory. It is a pretty rare case that they are different but it is possible.

5) *maturityDate* is the date on which the redemption is paid. In case of a coupon bond the last coupon is usually also paid on maturity date. Note that the interest is usually accrued for the settlement day but not for the maturity day!

Further parameters we need are not the member variables<sup>30</sup> of the *ZeroCouponBond* class, yet we need them to compute the bond yield.

6) *cleanPrice* is (in case of a zero bond) just its market value. In case of a coupon bond one needs to distinguish between clean and dirty price due to partly accrued but not yet paid coupon. Clean price is more comfortable because it does not jump on the coupon payment dates.

(http://en.wikipedia.org/wiki/Delivery\_versus\_payment)

<sup>&</sup>lt;sup>28</sup> Though this is hardly relevant for the financial calculation I do recommend you to read about settlement risk (http://en.wikipedia.org/wiki/Settlement\_risk) and delivery versus payment

<sup>&</sup>lt;sup>29</sup> Tag der Deutschen Einheit (German Unity Day). Interestingly, it is a business day for FrankfurtStockExchange, Xetra, Eurex, Euwax... though as far as I remember there is no trading activity on these exchanges on the 3rd of October.

<sup>&</sup>lt;sup>30</sup> Note that contrast to *ZeroCouponBond* the class *FixedRateBond* does have the member variable *dayCounter\_*... though it seems to be unused; at least neither doxygen call graph, nor Visial Assist "find reference", nor Visual Studio "call hierarchy" tells us where the method *FixedRateBond::dayCounter()* is called or where *dayCounter\_* is directly addressed.

7) *dayCounter*, as its name implies, counts the number of [interest] days <u>and</u> the year fraction between two dates. On one hand we said that a day is the basic time <u>span</u> for which the interest accrues. On the other hand a year is the basic time <u>unit</u> for the interest rate arithmetic.

In our cased we used ActualActual::ISDA. It works as follows: first of all it splits the time span by years: in our example to [7.10.2014, 31.12.2014]  $\cup$  [1.01.2015, 31.12.2015]  $\cup$  [1.01.2016, 4.10.2016). There are 86 days in [7.10.2014, 31.12.2014] and 2014 is not a leap year, so we divide 86 by 365 to get the year fraction. The year 2015 is completely within the time span, so it gives the year fraction equal to 1.0 (and it does not matter whether it a leap year or not). Finally, there are 277 interest<sup>31</sup> days in [1.01.2016, 4.10.2016) and 2016 is a leap year. So we divide 277 by 366 and the whole time span is equal to 86/365 + 1.0 + 277/366 = 1.9924 years.

There are pretty many day counters, have a look at *ql/time/daycounters*. Their logic is pretty straightforward and you can readily understand it from the source code and comments.

But there is one nuance: in case of the ActualActual::ISMA the year fraction additionally depends on the the reference period! That's why the method *yearFraction* in the basic abstract class *DayCounter* is declared as follows:

8) compounding means the interest rate compounding rule. QuantLib supports Simple (1 + rt), Compounded  $(1 + r)^t$ , Continuous  $\exp(rt)$  and SimpleThenCopounded rules. Compounded is the most common rule in Germany.

9) frequency means something periodical, e.g. how often the coupons are paid. A zero bond has no coupons but still the frequency may be relevant because by Compounded rule we actually have<sup>32</sup>  $(1 + \frac{r}{\text{freq}})^{t \cdot \text{freq}}$  Nice to know, since it is not mentioned in ql/compounding.hpp!

10) *accuracy* and *maxEvaluations* determine the numeral precision of the bond yield. In Germany it is required to calculate with 1.0e-6 precision but since in our case we check the yield by means of a closed form solution, we set very high precision to check whether the numerical iteration method can achieve it (it can).

After we calculated the bond yield with QuantLib business logic, we check it manually. Note that we add one day in year 2014 and do not do it in year 2016 because the settlement day is an interest day but the maturity day is not.

Also note the assert(abs(t - tDC) < accuracy) statement, it is very practical to use it if you expect that a certain condition in your program <u>always</u> takes place.

You might be quite confused with what you have just learnt: compounding rules, day counters, frequencies... At least I was very confused as I confronted them - and not due to the fact that the reality is more complicated than  $\exp(rt)$  but rather because there are so many of them. But do not panic! First of all, you will hardly need to learn all of them in detail. Actually, they are already implemented in QuantLib and you just need to understand how to use them properly. And you can always have a look at http://www.opengamma.com/sites/default/files/interest-rate-instruments-and-market-conventions.pdf

<sup>&</sup>lt;sup>31</sup> Recall that the last day (i.e. maturity day) is not an interest day.

<sup>&</sup>lt;sup>32</sup> Lines 53-54 from ql/interestrate.cpp

case Compounded: return std::pow(1.0+r\_/freq\_, freq\_\*t);

This excellent reference "contains what everybody is supposed to know when they first start working in the industry".

Our case with a zero bond is not only an excellent opportunity to overview the realm of the market conventions. It is also good for discussing the [limitations of] second basic OOP principle, the inheritance. In some introductory books on OOP the inheritance is interpreted as possibility to <u>evolve from simple to complex</u>, starting with a very basic [abstract] class and making derived classes functionality richer and richer. Some methods are declared but not implemented in basic classes and it is the task of the derived classes to have them implemented. If necessary, the derived class may also <u>override</u> the implementation of the basic class(es) methods. In a sense we can see it at Figure 1.7. The basic "financial" class is the abstract class *Instrument*. This class is not the first in class hierarchy, it inherits from *LazyObject*, which in turn inherits from *Observer* and *Observable*.

*LazyObject* is a *design pattern* with motto "(re)calculate only when you need it". The goal is the optimization of the computational performance (e.g. it may be very costly to recalculate 100000 Monte Carlo paths). *Observer-Observable* design pattern is as follows: when an *Observer* changes its state, it notifies all its *Observables*; in this case the *Observables* (that are often *LazyObjects*) usually need to recalculate themselves.

The *Observer, Observable* and the *LazyObject* classes implement purely technical features, which (according to the philosophy of QuantLib) each financial instrument should possess in order to make financial computations faster and more reliable from the technical point of view. As we have already mentioned, "financial hierarchy" begins with *Instrument*. It has some very basic properties (and respective methods) that each financial instrument should have: e.g. every (traded) instrument should obviously have an NPV (net present value). But e.g. the calculation of a bond NPV is very different from the calculation of an option NPV (and yet both are financial instruments). Thus *Instrument* class "is purely abstract<sup>33</sup> and defines the interface of concrete instruments which will be derived from this one"<sup>34</sup>.

Respectively, *Bond* inherits all methods from *Instrument* (and implements some of them). Additionally, it has some bond-only specific methods. So far we can see a perfect realization of the "from simple to complex" paradigm. However, there are some problem in this sense with *ZeroCouponBond*. On the one hand a *ZeroCouponBond* is a special case of a [coupon] *Bond* (which is fully in accordance with class hierarchy that we see at Figure 1.7). One the other hand the functionality of a *ZeroCouponBond* is much simple than that of a [coupon] *Bond*.

What does it all mean? Just the fact, that there is not perfect programming paradigms that perfectly cover all situations in real life. But it also means that inheritance paradigm is sometimes misunderstood. "From simple to complex" is not the only goal of inheritance. Actually, some experts recommend to replace inheritance with encapsulation because long inheritance paths makes program less readable (e.g. have a look at the Doxygen class diagram for *CallableZeroCouponBond*). Additionally, it becomes more and more difficult to maintain logically consistent class hierarchy, e.g. why<sup>35</sup> both *FixedRateBond* and *ZeroCouponBond* are derived directly from *Bond* but *CallableZeroCouponBond* is derived from *CallableFixedRateBond*, which in turn is derived from *CallableBond*.

<sup>34</sup> Cited from the class description in the source code of Instrument.hpp

<sup>&</sup>lt;sup>33</sup> //! Abstract instrument class

<sup>/\*!</sup> This class is purely abstract and defines the interface of concrete instruments which will be derived from this one. Strictly speaking, from the technical point of view this class is not purely abstract since some methods, e.g. setPricingEngine() are implemented. But from the business logic point of view it may be called purely abstract.

<sup>&</sup>lt;sup>35</sup> Probably because *CallableZeroCouponBond* inherits the overridden method setupArguments() from *CallableFixedRateBond*. But this is a weak argument since one could have implemented this methods directly in *CallableBond* because *CallableFixedRateBond* is the only class, which inherits from *CallableBond*.





But inheritance (together with the 3rd OOP principle <u>polymorphism</u>) has another important goal (which often remains unexplained in introductory books): <u>programmers must be able to reduce the objects to the properties and methods of the [very] basic classes</u>. Figure 1.8 shows how many classes inherit from *Instrument*. Now imagine that we need to build a risk modeling engine on the basis of QuantLib in order to simulate the evolution of our portfolio. In this context it does not matter which instruments we have in our portfolio. All we need to model is the evolution of NPV. Thanks to polymorphism we need not to distinguish between the dozens of classes but can just call the NPV related methods that are <u>declared</u> in *Instrument*. A compiler will by itself call the proper <u>implementations</u> of these methods (that may be different in derived classes). But in order to do this a compiler needs an inheritance hierarchy<sup>36</sup>.

Finally, in order to complete the discussion of the [basics of] date arithmetic, I highly recommend you to have a look at the blog post "Odds and ends: date calculations" by Luigi Ballabio: http://implementingquantlib.blogspot.de/2014/11/odds-and-ends-date-calculations.html In particular, this post discusses *Periods* and *Schedules*.

Periods are not as straightforward as they may seem to be at first glance but the nuances are mostly technical. In either case, at least at first approximation Periods may be comprehended as counterparts of Frequencies (which we have already encountered). In particular, the Periods are used in such a way to construct Schedules. Luigi writes in his blog post, "Schedule class ... is used to generate sequences of coupon dates". This is definitely the most frequent but not the only usage of *Schedule*. Have a look at Doxygen caller graphs for the methods of *Schedule* and you will see that it is used by *MarketModel*, *CDO*, *CPISwap*.

Schedule is thoroughly explained by Dimitri Reiswich: http://quantlib.org/slides/dima-ql-intro-1.pdf (Slide 32). Let us construct a schedule and then use it to create a coupon bond.

The section "bond parameters" is already familiar to us from the Code 1.2. There is a new parameter *accrualConvention*, which is set to *Unadjusted*. A schedule will generate a sequence of the coupon periods. But if the begin or the end of a such period falls on a holiday, shall we shift it (somehow) to a working day? At the first glance we shall not because, as we have already mentioned, all calendar days are interest days. On the other hand the payment will be shifted (since one can hardly commit a financial transaction on a holyday). In our example the end of the last coupon period is 02.10.2016 (Sunday) and 03.10.2016 is a holiday, so this coupon will be paid on 04.10.2016. But this means that no interest will be accrued for 02.10.2016 and 03.10.2016. Setting *accrualConvention* equal to *paymentConvention* would fix this problem ... but there is no 100% ubiquities convention how to relate coupon periods and coupon payments. That's the business day conventions are set separately for both of them.

The next section makes to schedules: making schedule is straightforward. Note that it is a common practice (at least in Germany) to generate payments backwards from the end date. Also note that that the Period is defined in terms of frequency.

In either case, the constructor has a lot of parameters and it might be not so easy to memorize all of them (at least it is not clear for what accrualConvention stands for two times). To solve this problem we may aply the <u>Named Parameter Idiom</u>. Technically it is implemented as follows: calling <u>MakeSchedule()</u> creates an object of class <u>MakeSchedule</u>. All other methods we call set a respective parameter and return the <u>reference</u> to the same object. For example the method *from* is defined as

<sup>&</sup>lt;sup>36</sup> Alternative approach in C++ is the template programming, which does not require a shared ancestor class. The question whether to choose inheritance or templates is pretty usually pretty straightforward: if there is an "obvious" class hierarchy then choose the former otherwise choose the latter. But in general you should never mix them! In chapter 3 we consider the class *PiecewiseYildCurve* and encounter such a mixture, which is pretty hard to understand.

```
MakeSchedule& MakeSchedule::from(const Date& effectiveDate) {
    effectiveDate_ = effectiveDate;
    return *this;
}
```

Note "&" in MakeSchedule& i.e. it returns a reference and also note that there is so far an object of *MakeSchedule*, not of *Schedule* class!

Finally, as the assignment operator (MakeSchedule::operator Schedule()) is called, it creates an object of class Schedule.

Then we just check that schedule and schedule2 are identical.

```
#include <iostream>
#include <ql/quantlib.hpp>
#include <math.h>
using namespace QuantLib;
void main()
{
       //bond parameters
       Date issueDate(2, October, 2014);
       Natural settlementDays = 2;
       Calendar calendar = Germany(Germany::Settlement);
       BusinessDayConvention paymentConvention = Following;
       BusinessDayConvention accrualConvention = Unadjusted;
       Real faceAmount = 100;
       Real redemption = 100;
       Date maturityDate(2, October, 2016);
       DayCounter dayCounter = ActualActual(ActualActual::ISDA);
       Frequency frequency = Annual;
       //make schedules via constructor
       Schedule schedule(issueDate, maturityDate, Period(frequency), calendar,
              accrualConvention, accrualConvention, DateGeneration::Backward, false);
       //... and via Named Parameter Idiom helper
       Schedule schedule2 = MakeSchedule().from(issueDate).to(maturityDate).
              withFrequency(frequency).withCalendar(calendar).
              withConvention(accrualConvention).
              withTerminationDateConvention(accrualConvention).
              backwards().endOfMonth(false);
       //check that the schedules are identical
       std::vector<Date> datesSchedule = schedule.dates();
       std::vector<Date> datesSchedule2 = schedule2.dates();
       assert(datesSchedule.size() == datesSchedule2.size());
       for (unsigned int d = 0; d < datesSchedule.size(); d++)</pre>
       {
              assert(datesSchedule[d] == datesSchedule2[d]);
std::cout << "Schedule Date " << d + 1 << " " << datesSchedule[d] << "\n";</pre>
       }
       //coupons
       std::vector<Rate> coupons(datesSchedule.size() - 1, 0.05);
       //bonds
       FixedRateBond couponBond(settlementDays, faceAmount, schedule, coupons,
              dayCounter, paymentConvention, redemption, issueDate, calendar);
```

```
FixedRateBond couponBond2(settlementDays, calendar, faceAmount, issueDate,
              maturityDate, Period(frequency), coupons, dayCounter, accrualConvention,
              paymentConvention, redemption, issueDate, Date(), DateGeneration::Backward,
              false, calendar);
       //bond cashflow
       Leg bondCashflow = couponBond.cashflows();
       for (unsigned int d = 0; d < bondCashflow.size(); d++)</pre>
              std::cout << "Payment " << d + 1 << ":</pre>
                                                         " << (bondCashflow[d])->date()
                     << " " << (bondCashflow[d])->amount() << "\n";
       //check that both bonds pay the same interest
       for (Date d = issueDate; d <= maturityDate; d++)</pre>
       {
              double accuredInterest = couponBond.accruedAmount(d);
              double accuredInterest2 = couponBond2.accruedAmount(d);
              std::cout << "Interest accrued on " << d << " " " << accuredInterest << "\n";</pre>
              assert(abs(accuredInterest - accuredInterest2) < 1.0e-15);</pre>
       }
Code 1.3 Constructing schedules and coupon bonds
```

Further we generate two coupon bonds, one with schedule we constructed before and one via constructor (which internally generates a schedule). Obviously, you should favor the former, since the process is split in two steps and you can check the schedule before creating a bond. As we have already noted, coupon payment dates and the end of coupon periods may differ.

Finally, we check that the interest is accrued identically by couponBond and couponBond2. There is a curios nuance<sup>37</sup>: in the constructor of the *Bond* class (which is parent for *FixedRateBond*) there is a following plausicheck:

```
if (issueDate_ != Date()) {
    QL_REQUIRE(issueDate_<cashflows_[0]->date(),
        "issue date (" << issueDate_ <<
        ") must be earlier than first payment date (" <<
            cashflows_[0]->date() << ")");
}</pre>
```

I would expect something like

QL\_REQUIRE(

```
calendar_.advance(issueDate_, settlementDays_, paymentConvention_) < cashflows_[0]->date()
)
```

i.e. that no payment are allowed earlier than settlement date. But it is not the case, moreover, you cannot easily hack the QuantLib and modify this plausi-check in such a way because paymentConvention\_ is not a member of *Bond* (and not even of *FixedRatebond*). Rather it is a member of Leg cashflows\_ (which, in turn, is a member of *Bond* but is initialized in the constructor of *FixedRateBond* <u>after</u> the call of the parent constructor; this it is impossible to get paymentConvention\_ within the constructor of *Bond* even if cashflows\_ would store the paymentConvention\_ as member variable (it does not)).

<sup>&</sup>lt;sup>37</sup> Skip it if is too hard to comprehend, it is just technical details that were curios for *me*.

## **Chapter 3: (Yield) Term Structures**

From the programmer's point of view the term structures are well-documented in Luigi Ballabio's book: https://dl.dropboxusercontent.com/u/13584583/qlbook/chapter3.pdf, which you are certainly recommended to have a look at. The most important term structures are the yield term structures. As Luigi himself states "it was even called TermStructure back in the day, when it was the only kind of term structure in the library and we still hadn't seen the world". In this chapter we will discuss the yield term structures by means of two practical examples.

Example 1: As I was a recent graduate I actively looked for job and once got the following exercise:

| Market Data 1 (maturity in years, annual coupon, annually compounded rates, face value 100) |       |       |       |       |       |       |       |       |       |        |        |
|---|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|
|   | Bond1 | Bond2 | Bond3 | Bond4 | Bond5 | Bond6 | Bond7 | Bond8 | Bond9 | Bond10 | Bond11 |
| Maturity  | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     | 10     | 9      |
| Coupon  | 4     | 5     | 5     | 4     | 5     | 6     | 6     | 6     | 6     | 5      | 3      |
| PV  | 101.5 | 103   | 102.5 | 98    | 101   | 105   | 104.5 | 104   | 104   | 96     | ???    |

| Market Data 2 (maturity in years, annual coupon, annually compounded rates, face value 100) |       |       |       |       |       |       |       |       |       |        |        |
|---|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|
|   | Bond1 | Bond2 | Bond3 | Bond4 | Bond5 | Bond6 | Bond7 | Bond8 | Bond9 | Bond10 | Bond11 |
| Maturity  | 1.7   | 2.1   | 3     | 3.8   | 4.6   | 5.6   | 6.5   | 7.2   | 8     | 9      | 5.4    |
| Coupon  | 4     | 5     | 4     | 5     | 4     | 6     | 4     | 5     | 4     | 5      | 4      |
| PV  | 104.2 | 110   | 103   | 105   | 100   | 103   | 96    | 103   | 92    | 98     | ???    |

Zero rate from 0 to 1 year is assumed constant and equal to 0.02

Question: What is the present value of bond 11 using a bootstrapping algorithm? The maturities m(i) of bonds 1 to 10 may be assumed to be increasing and chosen so that m(i+1)-m(i)<=1 Please use annually compounded rates. Both Excel-based and VBA-based solutions are accepted.

The first part is pretty straightforward, such exercises are done by virtually any MFE Program. First of all we find the yield of the first bond  $r_1$ . It holds

$$rac{ ext{coupon}_1 + ext{FaceValue}_1}{1 + r_1} = PV_1$$
 i.e.  $rac{100 + 4}{1 + r_1} = 101.5$  so  $r_1 = 0.024630$ 

Further we have

$$\frac{\text{coupon}_2}{1+r_1} + \frac{\text{coupon}_2 + \text{FaceValue}_2}{(1+r_2)^2} = PV_2$$
 i.e.  $\frac{5}{1.024630} + \frac{100+5}{(1+r_2)^2} = 103.0$ 

Thus

$$(1+r_2)^2 = 105/98.120190 = 1.070116$$

and respectively

$$r_2 = \exp(\ln(1.070116)/2) - 1 = 0.034464$$

It is the so-called *bootstrapping* algorithm, which can be continued iteratively. Of course we need not solve it manually, rather we can solve it like in Code 3.1, for which we even do not need QuantLib. The only problem is that it is never the case in real world.

The second case is much more realistic. However, there is no non-ambiguous algorithm to solve it. Obviously, we need to interpolate but what - yields, discount factors or something else? And of course there is no the only right way to connect the dots. As I tried to solve this problem I tried several plausible approaches but none of them has satisfied my interviewer. Then he gave me a hint to "use linear

interpolation" (of what?!) ... I did my best but he still was not satisfied. I think I still would not satisfy this guy if I had solved the problem with QuantLib ... but probably I could have impressed him with the knowledge of common term structure fitting approaches (that are rarely learnt in detail by MFE programs).

```
#include <iostream>
#include <math.h>
void main()
{
  //market data
  double coupons[] = { 4.0, 5.0, 5.0, 4.0, 5.0, 6.0, 6.0, 6.0, 6.0, 5.0 };
  double pvs[] = { 101.5, 103.0, 102.5, 98.0, 101.0, 105.0, 104.5, 104.0, 104.0, 96.0 };
  const double FACE VALUE = 100.0;
 //iteratively calculate yields by bootstrapping algorithm
  const int n = sizeof(coupons) / sizeof(coupons[0]);
  double yields[n];
  yields[0] = (FACE VALUE + coupons[0]) / pvs[0] - 1;
  std::cout << "yield 1: " << yields[0] << "\n";</pre>
  for (int i = 1; i < n; i++)</pre>
  {
       double couponsPV = 0.0;
       for (int c = 1; c <= i; c++)</pre>
       {
           couponsPV += coupons[c] / pow((1.0 + yields[c - 1]), c);
       }
       yields[i] = exp(log((FACE_VALUE + coupons[i]) / (pvs[i] - couponsPV)) / (i + 1)) - 1;
       std::cout << "yield " << i + 1 << ": " << yields[i] << "\n";</pre>
  }
  //answer the question
  const double bond11Coupon = 3.0;
  const int bond11Maturity = 9;
  double bond11PV = 0.0;
  for (int i = 1; i <= bond11Maturity; i++)</pre>
  {
      bond11PV += bond11Coupon / pow((1.0 + yields[i - 1]), i);
  }
  bond11PV += FACE_VALUE / pow((1.0 + yields[bond11Maturity - 1]), bond11Maturity);
  std::cout << "PV of the Bond11 is " << bond11PV << "\n";</pre>
}
```

#### Code 3.1 Bootstrapping algorithm in an ideal case

Let us have a look at Figure 3.1 to find a suitable tool for our task. At first glance it seems that we need to take *FittedBondDiscountCurve*<sup>38</sup>. There is a small nuance with *FittedBondDiscountCurve*, it contains a class *FittingMethod*, which is a base class for several fitting methods, as Figure 3.2 shows. But this nuance is not a problem, the specification of the *FittingMethod* is straightforward and Google readily helps to grasp the ideas behind the fitting methods.

But there is also a big nuance: Figure 3.1 does not give us the full picture! There is class *QuantLib*:: *PiecewiseYiledCurve*, which allows us to adhere to the hint to "use linear interpolation"<sup>39</sup>, whereas

<sup>&</sup>lt;sup>38</sup> Note that though we can easily calculate the yields of our bonds, the *ZeroYieldStructure* will not do because we have coupon bonds, not zero bonds.

<sup>&</sup>lt;sup>39</sup> An advanced quant developer (like me <sup>(2)</sup>) often confronts the following problem: he sees that the software requirements he needs to implement are actually suboptimal. But unauthorized deviations from the requirements are

*FittedBondDiscountCurve* does not allow it. But it seems that *PiecewiseYiledCurve* does <u>not</u> inherit from *YieldTermStructure*! Indeed it does, but in a pretty complicated manner. As we can see from Figure 3.1a, *PiecewiseYildCurve* intensively (ab)uses templates. Luigi explains this complex construction in his book in detail. Unless you are really curious, you do not need to spend your time on them. Just to put it briefly: *PiecewiseYildCurve* inherits from *Traits::template curve<Interpolator>::type*, which is a template class. Meaningful Traits are defined in /ql/termstructures/yield/bootstraptraits.hpp, there are (currently) three structures: *Discount, ZeroYield* and *ForwardRate*.





usually unwelcome. So in this case the best solution would be to implement both the required approach and a better alternative (explaining, why the latter is better).

In turn, each Trait-structure contains a struct curve, e.g. in case of *Discount* it is struct curve {

typedef InterpolatedDiscountCurve<Interpolator> type;

};

and *InterpolatedDiscountCurve<Interpolator>* does inherit from *YieldTermStructure* (see Figure 3.1). Thus *PiecewiseYildCurve* inherits from *YieldTermStructure* via Traits. But *Traits* is a template class and as long as it is unspecified, the inheritance cannot be recognized.



Code 3.2 is in a sense a slightly adopted (but thoroughly explained) version of the code from **FittedBondCurve** (a sample project from the QuantLib suite).

Code 3.2 brings us a lot of interesting features. First of all, we should clarify whether the bond prices are clean or dirty<sup>40</sup>. Since the former is usually the case, we assume it (under this assumption it is also somewhat easier to work with QuantLib). Further do not be confused with numberOfBonds = 11. The first bond (with coupon of 0.02 and maturity of 1.0) just depicts the statement " Zero rate from 0 to 1 year is assumed constant and equal to 0.02".

Further we confront something new: std::vector< boost::shared\_ptr<SimpleQuote> > quote; .

*SimpleQuote* (which inherits from the abstract basis class *Quote*) is just a structure to keep (whatever) market quotes. It contains a double value and a bool flag whether a quote is still valid (you know, the market quotes: prices, implied volatilities, interest rates and so on change quickly <sup>(i)</sup>).

As to *boost::shared\_ptr<T>*, it is a <u>shared</u> pointer<sup>41</sup> (a member of <u>smart</u> pointers family), a very practical programming trick. Actually, it automates the memory management (to some extent it resembles Java references). The implementation of the shared pointers in boost is complicated but essentially its member variables are T \* px (a conventional pointer) and the reference counter *boost::detail::shared\_count pn*. Every time a shared pointer is passed as an argument, the reference counter is increased by one. The data, to which *px* points to is allocated in heap (since *px* is just a normal pointer).

<sup>41</sup>Dimitri Reiswich provides a concise hands-on introduction to boost shared pointers:

http://quantlib.org/slides/dima-boost-intro.pdf

<sup>&</sup>lt;sup>40</sup> In previous task it was not an issue since there were no fractional maturities and thus no fractional coupons.

Also recall that we setup Natvis visualizer for boost by Arkady Shapkin, in partucular, they contain visualizers for shared pointers.

```
#include <iostream>
#include <ql/quantlib.hpp>
#include <math.h>
using namespace QuantLib;
#define LENGTH(a) (sizeof(a)/sizeof(a[0]))
void main()
{
       const Size numberOfBonds = 11;
       Real cleanPrices[] = { 100.0, 104.2, 110, 103, 105, 100, 103, 96, 103, 92, 98};
       Real lengths[] = { 1.0, 1.7, 2.1, 3, 3.8, 4.6, 5.6, 6.5, 7.2, 8, 9 };
       Real coupons[] = { 0.02, 0.04, 0.05, 0.04, 0.05, 0.04, 0.05, 0.04, 0.05, 0.04, 0.05 };
       const double FACE VALUE = 100.0;
       const double COUPON = 4.0;
       const double MATURITY = 5.4;
       std::vector< boost::shared_ptr<SimpleQuote> > quote;
       for (Size i = 0; i < numberOfBonds; i++) {</pre>
              boost::shared_ptr<SimpleQuote> cp(new SimpleQuote(cleanPrices[i]));
              quote.push back(cp);
       }
       RelinkableHandle<Quote> guoteHandle[numberOfBonds];
       for (Size i = 0; i < numberOfBonds; i++) {</pre>
              quoteHandle[i].linkTo(quote[i]);
       }
       Frequency frequency = Annual;
       DayCounter dc = SimpleDayCounter();
       BusinessDayConvention accrualConvention = ModifiedFollowing;
       BusinessDayConvention convention = ModifiedFollowing;
       Real redemption = 100.0;
       Calendar calendar = NullCalendar();
       Date today = calendar.adjust(Date::todaysDate());
       Date origToday = today;
       Settings::instance().evaluationDate() = today;
       Natural bondSettlementDays = 0;
       Natural curveSettlementDays = 0;
       Date bondSettlementDate = calendar.advance(today, bondSettlementDays*Days);
       std::cout << "Today's date: " << today << std::endl;</pre>
       std::cout << "Bonds' settlement date: " << bondSettlementDate << std::endL;</pre>
       std::vector<boost::shared ptr<BondHelper> > instrumentsA;//for FittedBondDiscountCurve
       std::vector<boost::shared ptr<RateHelper> > instrumentsB;//for PiecewiseYieldCurve
       for (Size j = 0; j < LENGTH(lengths); j++) {</pre>
              Date maturity = bondSettlementDate;
              while (dc.yearFraction(bondSettlementDate, maturity) < lengths[j]) {</pre>
                     maturity++;
              }
              Schedule schedule(bondSettlementDate, maturity, Period(frequency),
                     calendar, accrualConvention, accrualConvention,
                     DateGeneration::Backward, false);
```

```
boost::shared_ptr<BondHelper> helperA(
            new FixedRateBondHelper(quoteHandle[j],
            bondSettlementDays,
            FACE_VALUE,
            schedule,
            std::vector<Rate>(1, coupons[j]),
            dc,
            convention,
            redemption));
      boost::shared ptr<RateHelper> helperB(
            new FixedRateBondHelper(quoteHandle[j],
            bondSettlementDays,
            FACE VALUE,
            schedule,
            std::vector<Rate>(1, coupons[j]),
            dc,
            convention.
            redemption));
      instrumentsA.push back(helperA);
      instrumentsB.push back(helperB);
}
bool constrainAtZero = true;
Real tolerance = 1.0e-10;
Size max = 5000:
boost::shared ptr<YieldTermStructure> ts0(
      new PiecewiseYieldCurve<ZeroYield, Linear>(curveSettlementDays,
      calendar.
      instrumentsB,
      dc));
double d5 4 = ts0->discount(MATURITY);
double d4 4 = ts0->discount(MATURITY - 1);
double d3_4 = ts0->discount(MATURITY - 2);
double d2_4 = ts0->discount(MATURITY - 3);
double d1 4 = ts0->discount(MATURITY - 4);
double d0 4 = ts0->discount(MATURITY - 5);
double f = MATURITY - 5; //fraction of the 1st coupon
double bond11PV
      = COUPON*(d5_4 + d4_4 + d3_4 + d2_4 + d1_4 + d0_4*f) + FACE_VALUE*d5_4;
std::cout << "PV Bond11 (<ZeroYield, Linear> interpolation): " << bond11PV << "\n";</pre>
//<Discount, LogLinear> is more common in practice than <ZeroYield, Linear>
boost::shared_ptr<YieldTermStructure> ts1(
      new PiecewiseYieldCurve<Discount, LogLinear>(curveSettlementDays,
      calendar,
      instrumentsB,
      dc));
d5 4 = ts1->discount(MATURITY);
d4_4 = ts1->discount(MATURITY - 1);
d3 4 = ts1->discount(MATURITY - 2);
d2 4 = ts1->discount(MATURITY - 3);
d1 4 = ts1->discount(MATURITY - 4);
d0 4 = ts1->discount(MATURITY - 5);
```

```
f = MATURITY - 5;
bond11PV = COUPON*(d5 4 + d4 4 + d3 4 + d2 4 + d1 4 + d0 4*f) + FACE VALUE*d5 4;
std::cout << "PV Bond11 (<Discount, LogLinear> interpolation): " << bond11PV << "\n";</pre>
boost::shared ptr<FittedBondDiscountCurve> ts2(
       new FittedBondDiscountCurve(curveSettlementDays,
       calendar.
       instrumentsA,
       dc,
       SvenssonFitting(),
      tolerance,
      max));
d5 4 = ts2->discount(MATURITY);
d4 4 = ts2->discount(MATURITY - 1);
d3 4 = ts2->discount(MATURITY - 2);
d2 4 = ts2->discount(MATURITY - 3);
d1 4 = ts2->discount(MATURITY - 4);
d0_4 = ts2->discount(MATURITY - 5);
f = MATURITY - 5;
bond11PV = COUPON*(d5 4 + d4 4 + d3 4 + d2 4 + d1 4 + d0 4*f) + FACE VALUE*d5 4;
std::cout << "PV Bond11 (Svensson interpolation): " << bond11PV << "\n";</pre>
/////The same via bond pricing engine/////////
RelinkableHandle<YieldTermStructure> yieldTermStructure;
boost::shared ptr<PricingEngine> bondEngine(
      new DiscountingBondEngine(yieldTermStructure));
Date maturity = bondSettlementDate;
while (dc.yearFraction(bondSettlementDate, maturity) < MATURITY)</pre>
                                                                    {
      maturity++;
}
Schedule schedule(bondSettlementDate, maturity, Period(frequency),
       calendar, accrualConvention, accrualConvention,
      DateGeneration::Backward, false);
FixedRateBond bond11(bondSettlementDays,
      FACE VALUE,
       schedule,
       std::vector<Rate>(5, COUPON/100.0),
       dc,
       convention,
      redemption);
bond11.setPricingEngine(bondEngine);
yieldTermStructure.linkTo(ts0);
std::cout << "PV Bond11 (pricing engine, ts0): " << bond11.NPV() << "\n";</pre>
yieldTermStructure.linkTo(ts1);
std::cout << "PV Bond11 (pricing engine, ts1): " << bond11.NPV() << "\n";</pre>
yieldTermStructure.linkTo(ts2);
std::cout << "PV Bond11 (pricing engine, ts2): " << bond11.NPV() << "\n";</pre>
```

**Code 3.2** Bootstrapping algorithms in a realistic case

}

But a surrounding smart pointer is allocated in stack, so its destructor is automatically called as the smart pointer gets out of scope. As soon as the destructor<sup>42</sup> is called, the reference counter is decreased by one. And if it was the last reference, a delete on px is committed<sup>43</sup>.

#### Further we see RelinkableHandle<Quote> quoteHandle[numberOfBonds].

Handle-Pattern is important concept in QuantLib<sup>44</sup>. Since it is much better to explain it by the example of term structure and pricing engine, we postpone it for a little bit.

Next new issue is Settings::instance().evaluationDate() = today;.

I believe, you have already got used to the fact that the advanced C++ features are intensively (ab)used it QuantLib: *evaluationDate()* returns a <u>reference</u> so the call of a this function on the left side of assignment expression is valid. In turn, <u>Settings::instance()</u>. implement the <u>Singleton</u> pattern, which means that there is no more than one instance of settings. If need not necessarily dwell upon this technical details, it suffices just to learn how the evaluation date is set.

Natural curveSettlementDays are the number of curve settlement days. It is analogous to bond settlement days that we have already considered in Code 1.2.

Further we see
std::vector<boost::shared\_ptr<BondHelper> > instrumentsA;//for FittedBondDiscountCurve
std::vector<boost::shared\_ptr<RateHelper> > instrumentsB;//for PiecewiseYieldCurve

As you readily see, both vectors are actually filled with [shared pointers to] the objects of *FixedRateBondHelper* class. A FixedRateBondHelper just contains the bond [data] and its [clean] price. A vector of FixedRateBondHelpers is used to calibrate a yield curve. But why do we need to distinguish between RateHelpers and BondHelpers (actually, *BondHelper* even inherits from *RateHelper*)?! The reason is very simple: the fact that *BondHelper* inherits from *RateHelper* does not imply that *std::vector<boost::shared\_ptr<BondHelper>>* inherits from *std::vector<boost::shared\_ptr<RateHelper>>*. And we pass the <u>vectors</u> of [shared pointers to] helpers as an argument for the constructors of curve builders.

Finally, we set up the yield curve. We start with PiecewiseYieldCurve<ZeroYield, Linear> since linear interpolation is required in task (and we can guess that the linear interpretation of zero yields is much more plausible than that of discount factors). After the curve is calibrated we can calculate the discount factors and price our bond. Since we calculate the clean price, we consider only a fraction of the first coupon.

We repeat the same calculation with PiecewiseYieldCurve<Discount, LogLinear> (because it is probably the most populat piecewise-interpolated term structure in practice) as well as with FittedBondDiscountCurve, SvenssonFitting (because Svensson's approach is used by German Central Bank<sup>45</sup>). As you see, the results are though similar but still different. And once again: there is no only right way to connect dots.

<sup>&</sup>lt;sup>42</sup> There is no destructor in shared\_ptr.hpp but *boost::detail::shared\_count pn* has a destructor, which is called in turn.

<sup>&</sup>lt;sup>43</sup> If you want to trace this process in detail, have a look at *BOOST\_SP\_INTERLOCKED\_DECREMENT* and

boost::checked\_delete

<sup>&</sup>lt;sup>44</sup> See also http://quantlib.org/slides/dima-ql-intro-1.pdf#Navigation81

<sup>&</sup>lt;sup>45</sup> We may have also used the cubic splines, like central US and Japanese banks do.

And additionally, we reproduce our calculation with DiscountingBondEngine. In QuantLib it is generally so that in order to price an instrument we need to attach a pricing engine to this instrument (we do it by means of bond11.setPricingEngine(bondEngine)). In turn, virtually every pricing engine needs a yield term structure. Or, better to say, a <u>Handle</u> to a term structure. Now we have a good opportunity to explain the Handle-Pattern.

In real world the yield change rapidly. And there may be hundreds of very different instruments and thus hundreds of pricing engines that, however, depend on the same term structure. Thus it would be too tedious to set a new term structure by each pricing engine.

Alternatively, we could make a term structure as a tuple of quotes and use Observer/Observable pattern. Of a quote changes, it notifies its Observer, i.e. a term structure, which in turn notifies its Observers, i.e. the pricing engines. Well, updating a dozen of quotes seems to be a better idea than updating of hundreds of engines but there is an even better idea. We introduce Handle, a mediator between the pricing engines and the term structure. A Handle notifies its pricing engine about updates of the term structure. In turn, this update mechanism (a.k.a. <u>relinking</u>) is implemented in <u>RelinakableHandle</u>, which is derived from Handle. Thus we create an RelinkableHandle, upcast it to Handle and set this Handle to the pricing engines. The engines will be notified of the term structure updates but they cannot relink the Handle to a new term structure. We, however, can (as soon as we keep an reference to the original RelinkableHandler). Actually, this is what we do in the last section of Code 3.2.

# **Chapter 4: Stochastic interest rate models**

In the previous chapter we discussed how to fit a yield curve to a set of bonds. This step, i.e. fitting the (current) term structure, is necessary for nearly every financial calculation. But this is, so to say, a static view. In turn, the stochastic interest rate models let us model the *dynamics* of the yield curve. For a (simple) risk management, we need to model the evolution of the interest rates under the real-world measure: e.g. if we want to calculate the VaR of a portfolio of (straight) bonds. But in order to price an interest rate derivative we need a model, calibrated under the martingale measure. Finally, there are the use cases in which we need the both measures: for instance, if a portfolio contains the interest rate derivatives and we want to calculate the VaR, we first simulate portfolio dynamics under the real-world measure (scenario generation) and then we switch to the risk-neutral measure in order to calculate the portfolio value for each scenario.

The most common in practice are the short rate models and the LIBOR market model. If you are looking for mathematical details you should read my tutorial, which is freely available in Internet<sup>46</sup>. Here I will just provide a quick summary. As is well known, in practice the shortest borrowing is the overnight borrowing (intraday borrowing are also possible but they are generally interest-free). It is also known that the central banks influence the short term rate but actually target the whole term structure. So the short rate models try to explain the evolution of the yield curve by the dynamics of the instantaneous spot rate (a.k.a. short rate).

But you should be careful: short rate is not an overnight rate! Rather it is an *unobservable* quantity, a mathematical abstraction. In particular, Damir Filipovic warns against a (straightforward) usage of the overnight rate time series for the calibration of a short rate model "because the motives and needs driving overnight borrowers are very different from those of borrowers who want money for a month or more"<sup>47</sup>. Model calibration is in general very challenging issue but in case of the short rate models it is particularly intricate and often insufficiently elucidated in MFE courses.

Usually one gets started with the Vasicek model

$$dr_t = (a - br_t)dt + \sigma dW_t$$

In original paper<sup>48</sup> Vasicek worked only with the real-world probability measures (in 1977 the risk-neutral pricing was not yet discovered). He constructed a risk-free portfolio (just like Black and Scholes did<sup>49</sup>) and used no arbitrage principle to derive the formula for a zero bond price. Later it was shown that the short rate dynamics under the martingale measure may be described with the same equation (with different drift but the same volatility). One can switch from the drift und the real-world measure to that under the martingale measure by means of the market price of risk  $\lambda$ .

So if there is a closed-form solution for the price of a zero bond can we calibrate the Vasicek model to the (current) yield curve? In principle, yes - we just need to choose the model parameters that minimize the distance<sup>50</sup> between the theoretical and market bond prices. Moreover, this will be a calibration directly under the risk neutral measure<sup>51</sup>. Note that in order to calibrate the stochastic *process* that drives the short

<sup>&</sup>lt;sup>46</sup> http://www.yetanotherquant.com/libor/tutorial.pdf

<sup>&</sup>lt;sup>47</sup> Damir Filipovic - Term Structure Models: A Graduate Course - Springer, 2009 (p. 10)

<sup>&</sup>lt;sup>48</sup> Notably, this paper is likely more often cited than read

<sup>(</sup>http://www.wilmott.com/messageview.cfm?catid=4&threadid=85948)

<sup>&</sup>lt;sup>49</sup> Actually, it was Merton, who did.

<sup>&</sup>lt;sup>50</sup> E.g. the sum of squares.

<sup>&</sup>lt;sup>51</sup> If you want to calibrate the model under the real-world measure you can either ignore Filipovic's warning and use a time series of the overnight rates as a proxy for the short rate or use more advanced approach like Kalman filter (see

rate one does not need a time series, rather a "static snapshot" (i.e. the current term structure) is enough. At the first glance it seems surprising but it is plausible: a bond price P(t,T) represents the expected evolution (und the martingale measure) of the short rate over the *whole* period (t,T).

However, the Vasicek model has a big problem, it is *not term structure consistent*, i.e. we cannot fit the model parameters in such a way that the empirical and theoretical bond prices coincide. Most likely this is the reason why Vasicek model is rarely used in practice. In particular, in QuantLib it seems to be just a stub for the Hull White model, which is term structure consistent. But before we discuss this model in detail it is worth getting a holistic view on models, implemented in QuantLib. Have a look at Figure 4.1. On the left from the dotted line there are "classical" models that are considered in the textbooks on term structure modeling<sup>52</sup>. The term structure consistent models are marked with **T**.

First, all (concrete) classes are derived from *QuantLib::CalibratedModel* which is obvious, since a noncalibrated model is useless in practice. But interestingly, the calibration logic is encapsulated in the basis class (except *QuantLib::MarkovFunctional*, where the method *calibrate()* is overridden).

Second, all short rate models except G2 are one factor. It is actually a pity because it is commonly agreed that one needs there factors in order to realistically model a yield curve: level, slope and curvature.

e.g.http://www.bankofcanada.ca/wp-content/uploads/2010/02/wp01-15a.pdf). Note that  $\lambda$  is not (directly) observable.

<sup>&</sup>lt;sup>52</sup> As to the right-hand side, have a look at http://quantlib.org/slides/qlws13/caspers.pdf



# Appendix A. Refactoring QuantLib::Date

Since it is very uncomfortable for debugging to work with serialNumber\_ representation of *Date*, we try to refactor this class. Since it would be very cumbersome to redesign it from scratch, our idea is as follows: we rename the current implementation of *Date* to *DateImpl* and then define a new class *Date* that will have all public methods the previous version of *Date* had and three member variables *Day day\_*, *Month month\_* and *Year year\_*. In all public methods we will create temporal instance(s) of *DateImpl*, call the respective methods of *DateImpl* and, if necessary, update the member variables of *Date*.

For example, the implementation of the method Date::month() will be as follows

```
Month Date::month() const
{
     DateImpl d(day_, month_, year_);
     return d.month();
```

Finally, we will write a respective natvis-Visualizer.

At the first glance it is a perfect idea: with little efforts we expect to get the <u>identical</u> functionality (probably, a little bit slow, but it does not matter). All we expect to do is to re-implement the public methods in a pretty trivial way, leaving the implementation of the previous *Date* (which is now *DateImpl*) intact. However, the practice is more complicated. First of all the files *date.hpp* and *date.cpp* contain not only the declaration and the definition of *Date* but also some auxiliary structures like *struct short\_date\_holder*, *struct long\_date\_holder*, etc (look for namespaces *io* and *detail*). They are used to print the objects of the *Data* class to console.

At the first glance there is no problem in context of our approach. However, the old version of *Date* (which is now *DateImpl*) uses the functionality of these auxiliary structures in a private method *DateImpl::checkSerialNumber(BigInteger serialNumber)*.

```
void DateImpl::checkSerialNumber(BigInteger serialNumber) {
    QL_REQUIRE(serialNumber >= minimumSerialNumber() &&
        serialNumber <= maximumSerialNumber(),
        "Date's serial number (" << serialNumber << ") outside "
        "allowed range [" << minimumSerialNumber() <<
        "-" << maximumSerialNumber() << "], i.e. [" <<
        Date::minDate() << "-" << Date::maxDate() << "]");
</pre>
```

In the last line we would, according to our idea, write Date**Impl**::minDate() << "-" << Date**Impl**::maxDate() but we cannot (or we need **additionally** define the output of *DateImpl* to the console)...

Well, using only *Date::minDate()* and *Date::maxDate()* in a private method of *DateImpl* is actually a small sin, which we can commit without remorse.

However, there is another problem: the range of valid dates is between 01.01.1901 and 31.12.2199 but there is a special case: the null date. It is created when one calls the default constructor *Date()*, which is now *DateImpl()*. One can apply the operators "=", "<", "<=", ">", ">=" to the null dates but cannot call the methods *dayOfYear()*, *month()*, *year()* without causing an exception.

```
Thus a nice attempt to re-implement e.g. the operator ">=" like this
```

```
inline bool operator>(const Date& d1, const Date& d2) {
    DateImpl di1(d1.dayOfMonth(), d1.month(), d1.year());
    DateImpl di2(d2.dayOfMonth(), d2.month(), d2.year());
    return (di1.serialNumber() > di2.serialNumber());
```

will fail because the attempt to call *dayOfYear()*, *month()*, *year()* in case of a null date throws an exception... A rude practical solution is to re-define the idea of the null date (actually, the default date). I decided to set the null date equal to the mindate, i.e. to 01.01.1901. This allows a straightforward re-implementation of all methods without a cumbersome consideration of the null date(s) case.

You need to put the file *QuantLib\_Date.natvis* to "\Documents\Visual Studio 2013\Visualizers"

| xml version="1.0" encoding="utf-8"?<br><autovisualizer quantlib::date"="" xmlns="http://schemas.&lt;br&gt;&lt;Type Name="><br/><displaystring>{{{day_}.{(int)month_}.{</displaystring></autovisualizer>  | microsoft.co<br>year_}} <th>m/vstudio/debugger/natvis/2010"&gt;<br/>splayString&gt;</th> <th></th> | m/vstudio/debugger/natvis/2010"><br>splayString> |              |
|--|--|--|--------------|
|  |  |  |              |
|  |  |  |              |
| Locals   |  |  | <b>-</b> ₽ × |
| Name   | Value  | Туре   | <u>_</u>     |
| vetAnotherVeryFirstDate  | {1.1.1901}   | QuantLib::Date                                   |              |
| veryFirstDate  | {1.1.1901}   | QuantLib::Date                                   |              |
| anotherVeryFirstDate   | {1.1.1901}   | QuantLib::Date                                   |              |
| 🔺 🤗 issueDate  | {2.7.2014}   | QuantLib::Date                                   |              |
| Gay_   | 2  | int  |              |
| Image: Second secon | July (7)   | QuantLib::Month                                  |              |
| 📽 year_  | 2014   | int  |              |
|  | -  |  |              |
|  |  |  | <b>T</b>     |
| Natvis specification for the refactored Q  | uantLib:Date   |  |              |

Finally, don't forget to run the QuantLib testsuite<sup>53</sup>. The refactoring case we have considered is an excellent justification of the unit test usefulness. The testsuite is pretty extensive, so if all tests run successfully we may be quite sure that our refactoring was correct. But since we changed the definition of the null date, a couple of tests do fail. Still these are very special cases, so we will further use the refactored version of *QuantLib::Date* in this book. But it is <u>not</u> recommended to use it in productive software!

# Wanna get the source code of the modified QuantLib:Date ?!

# Donate \$10 to the author ;)

<sup>&</sup>lt;sup>53</sup> NB! I may take a couple of days to complete!