Computational Finance with C++

Lecture 2: Introduction to Objected Oriented Programming

Panos Parpas
Imperial College London
p.parpas@imperial.ac.uk

Outline

- Objected Oriented version of Binomial Pricing Procedure
- Classes in C++
- Inheritance
- Virtual Functions
- Reading:
    - ✦Chapter 2 Capinski+Zastawniak,Numerical Methods in finance with C++

Binomial Model Class:
`BinModel02.h, BinModel02.cpp`

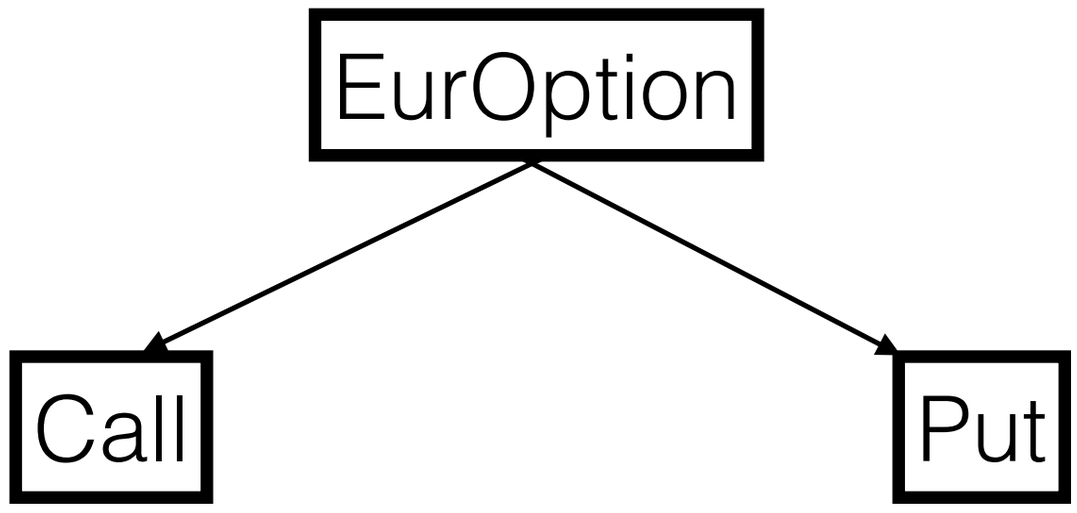
Things to note:

- Class declaration
- Private and public members
- Protected members
- Using the object-oriented version: `Options04.h, Options04.cpp and Main08.cpp`

Inheritance: `Options05.cpp, Main09.cpp`

Things to note:

- Classes and subclasses
- A new class can be derived from an existing class.
- The new class is called derived class “inherits” all members of existing (base) class.
- Derived class behaves as a subtype of the base class
- Object of the derived class can be assigned to a variable of the base class via pointers and references
- Constructors
- Classes and subclasses



Exercise

Exercise 2.1: A definite integral can be computed numerically by the trapezoidal approximation

$$\int_a^b f(x)dx \approx \frac{h}{2}(f(x_0) + 2f(x_1) + \ldots + 2f(x_{N-1}) + f(x_N))$$

where $h = \dfrac{b-a}{N}$ and $x_n = a + nh$ for $n = 0, 1, \ldots, N$.

Write a class called DefInt to compute the trapezoidal approximation for a given function f. The class should contain the following:

1. Private members to hold the values of the integration limits alb and a pointer to the function f.
2. A constructor function such that the integration limits a,b and the pointer to the funct f can be initiated at the time of creating an object of the class like this: DefInt MyInt(a,b,f)
3. A public function ByTrapezoid() taking N as an argument and returning the trapezoi approximation of the integral when called by MyInt.Trapezoidal(N);
4. You may also want to include another public function BySimspon() to compute the Simpson approximation to the integral (wikipedia link)

Virtual Functions: `Options06.h, Options06.cpp`

Things to note:

- Virtual functions are declared in a base class and may be redefined (overridden) in each derived class.
- They have the same name and set arguments in both base and derived class
- Can be used to clean design of function pointers
- Abstract classes
- Pure virtual functions
- Makes it easy to add new types of payoffs, e.g. review code `DoubDigitOpt.h` and `DoublDigitOpt.cpp`

Exercise 2.2: Add the ability to price bull and bear spreads by introducing new subclasses BullSpread and BearSpread of the EurOption class defined in Options06.h. Use the new classes, making subtle changes in Main10.cpp to price European bull and bear spreads.

The payoffs of a bull spread and a bear spread, which depend on two parameters $K_1 < K_2$ are given by

$$h^{bull}(z) = \begin{cases} 0 & \text{if } z \leq K_1 \\ z - K_1 & \text{if } K_1 < z < K_2 \\ K_2 - K_1 & \text{if } K_2 \leq z \end{cases}$$

$$h^{bear}(z) = \begin{cases} K_2 - K_1 & \text{if } z \leq K_1 \\ K_2 - z & \text{if } K_1 < z < K_2 \\ 0 & \text{if } K_2 \leq z \end{cases}$$

Exercise 2.3: Rewrite the code for numerical integration in Exercise 2.1 replacing function pointers by virtual functions.

Exercise 2.4: Add further payoffs, namely strangle and butterfly spreads by means of subclasses of the EurOption class placed in separate files, without changing anything in the existing code, except for the necessary changes in Main11.cpp to price options with the new introduced payoffs.

The payoffs of strangle and butterfly spreads, depend on two parameters $K_1 < K_2$ and are as follows:

$$h^{strangle}(z) = \begin{cases} K_1 - z & \text{if } z \leq K_1 \\ 0 & \text{if } K_1 < z \leq K_2 \\ z - K_2 & \text{if } K_2 < z \end{cases}$$

$$h^{butterfly}(z) = \begin{cases} z - K_1 & \text{if } K_1 < z \leq \frac{K_1+K_2}{2} \\ K_2 - z & \text{if } \frac{K_1+K_2}{2} < z \leq K_2 \\ 0 & \text{otherwise} \end{cases}$$