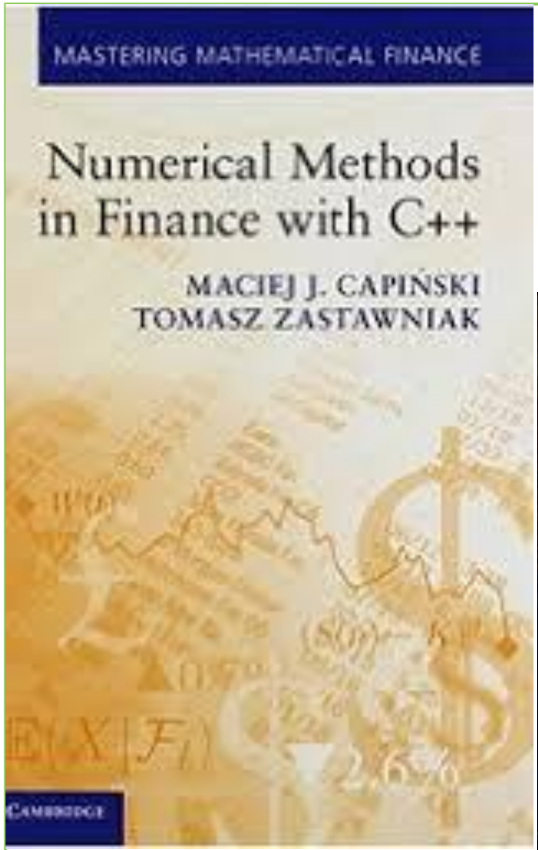
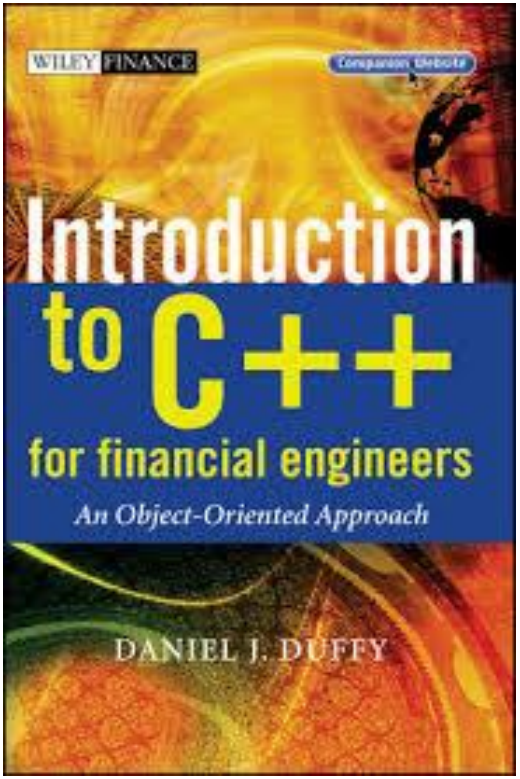
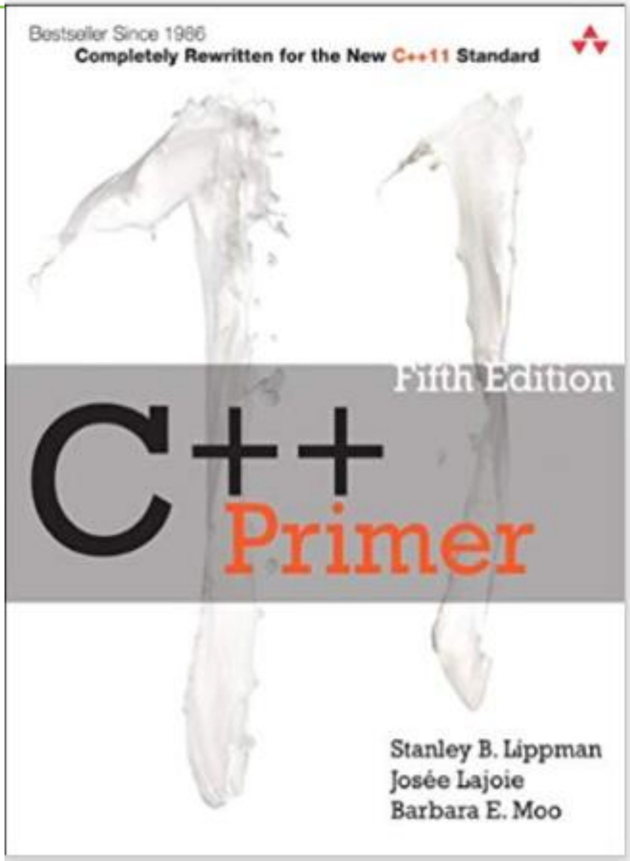
NYU | TANDON SCHOOL OF ENGINEERING

Topic 2
Structured Programming – Binomial Tree Model Implementation


MASTERING MATHEMATICAL FINANCE
Numerical Methods in Finance with C++
MACIEJ J. CAPIŃSKI
TOMASZ ZASTAWNIAK
CAMBRIDGE


WILEY FINANCE
Companion Website
Introduction to C++ for financial engineers
An Object-Oriented Approach
DANIEL J. DUFFY


Bestseller Since 1986
Completely Rewritten for the New C++11 Standard
Fifth Edition
C++ Primer
Stanley B. Lippman
Josée Lajoie
Barbara E. Moo

The Structured Programming

- Structured programming is a technique which arose from the analysis of the flow control structures which underlie all computer programs. It is possible to construct any flow control structure from three basic structures: sequential, conditional and iterative. We will use the structured programming to price European options via the binomial model.

Our first C++ program

```
#include <iostream>
using namespace std;
int main()
{   //display message
    cout << "Hello World!" << endl;
    // take input from keyboard
    double price = 0.0;
    cin >> price;
    cout << "price = " << price << endl;
    char x = '\0';  // null character
    cin >> x;
    cout << "x = " << x << endl;
    return 0;
}
```

- <iostream> header file for input and output
- namespaces std
- main function
- comments
- cin, cout and endl
- variables

```
/*
Hello World!
23.45
price = 23.45
A
x = A
*/
```

Binomial Option Pricing in C/C++

- The Structured Programming
  - Binomial Tree Model
    - Based on Function, Array and Pointer
  - CRR Option Pricer
    - Based on Function Call and Function Pointer
- The Object-Oriented Programming in C++
  - Binomial Tree Model Class
    - Based on C++ class
  - Option Pricer Framework
    - Based on Inheritance and Polymorphism

Binomial Tree Model

$$S(n, i) = S(0)(u)^i(d)^{n-i}$$

at step n and node i,

where S(0) > 0, u > d > 0

and n >= i >= 0

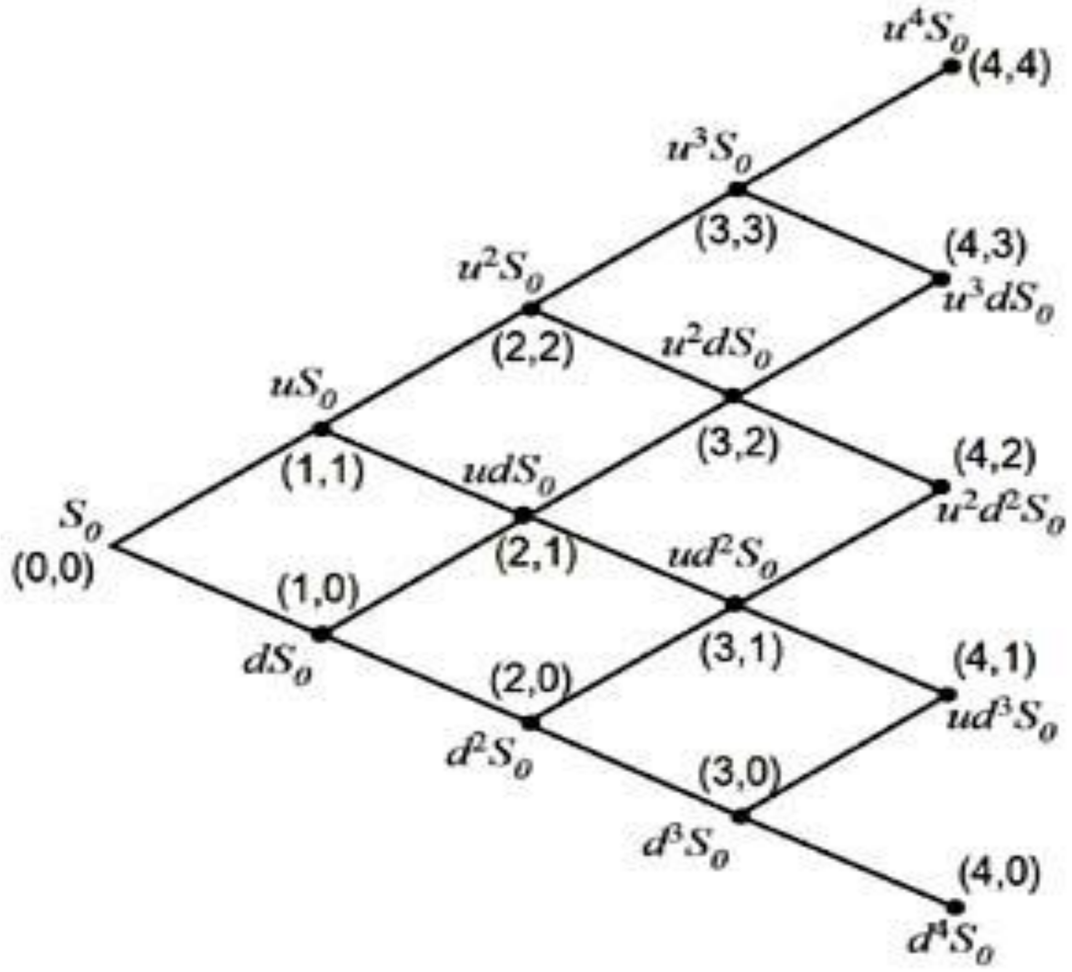


Figure 1. Binomial Tree Structure

Cox-Ross-Rubinstein (CRR) procedure

- At the expiry date N, H(N, i) = h(S(N,i)), for each node i = 0, 1, …, N
- If H(n+1, i) is already known at each node i = 0, 1, …, n+1 for some n = 0, 1, …N-1, then for each i = 0, 1, …,n

$$H(n,i) = \frac{qH(n+1,i+1)+(1-q)H(n+1,i)}{R}$$

- q = (R-D)/(U-D) is the risk-neutral probability
- The payoff functions

$$h^{call}(z) = \begin{cases} z-K \; if \, z > K \\ 0 \; otherwise \end{cases} = (z-K)^+ \qquad h^{put}(z) = \begin{cases} K-z \; if \, z < K \\ 0 \; otherwise \end{cases} = (K-z)^+$$

Implement Binomial Tree Model

- Using **binomial tree model** for asset pricing

- Compute and display the stock price
  - $S(n,i) = S(0)(u)^{i}(d)^{n-i}$

- cout << "S(n,i) = " << S0*pow(u,i)*pow(d,n-i)<< endl;

BinomialTreeModel01.cpp

```
// Calculate asset price at a specific node on the Binomial Tree
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()  {
double u = 1.15125, d = 0.86862;
double s0 = 106.00;
// compute asset price at node n = 3, i = 2
int n = 3;
int i = 2;
cout << "Asset Price at Binomial Tree Node(" << n << "," << i << ") = " << fixed << setprecision(2);
cout << s0 * pow(u, i) * pow(d, n - i);
cout << endl;
return 0;
}
/*
Asset Price at Binomial Tree Node(3,2) = 122.03
*/
```

What we learned from BinomialTreeModel01.cpp:

- #include <iomanip>
  - << fixed << setprecision(2);
- #include <cmath>
  - pow(u, i) * pow(d, n - i);
- How to compute asset price at a node entered by user?
  - int n = 0;
  - int i = 0;
  - cout << "Enter values for n and i: ";
  - cin >> n >> i;
- How to compute asset price at **Every Node** on the Binomial Tree?

BinomialTreeModel02.cpp

```
// Calculate asset price at every node on the Binomial Tree
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
        double u = 1.15125, d = 0.86862;
        double s0 = 106.00;
        for (int n = 0; n <= 8; n++)
        {
                for (int i = 0; i <= n; i++)
                {
                        cout << "Asset Price at Binomaial Tree Node(" << n << "," << i
                        << ") = " << fixed << setprecision(2);
                        cout << s0 * pow(u, i) * pow(d, n - i) << endl;
                }
        }
        return 0;
}
```

What we learned from BinomialTreeModel02.cpp:

- Nested for loops:
  - for (int n = 0; n <= 8; n++)
    - for (int i = 0; i <= n; i++)
- The Big-O notation:
  - express the upper bound of the runtime of an algorithm and thus measure the worst-case time complexity of an algorithm.
- Increment or decrement:
  - i++, i--, ++i, --i
- How could we store the asset price values from the Binomial Tree?

BinomialTreeModel03.cpp

```
// Use one-dimensional array to hold asset price
// at every node on the Binomial Tree
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
const int SIZE = 81;
int main()
{
        double u = 1.15125, d = 0.86862;
        double s0 = 106.00;
        double aPrice = 0.0;
        double prices[SIZE];
        for (int i = 0; i < SIZE; i++)
                prices[i] = 0.0;
```

```
// Compute asset price at every node on the Binomial Tree
// and store in the price array
int index = 0;
for (int n = 0; n <= 8; n++)
{
        for (int i = 0; i <= n; i++)
        {
                aPrice = s0 * pow(u, i) * pow(d, n - i);
                prices[index++] = aPrice;
        }
}
```

```
// Print out the value in the price array
index = 0;
for (int n = 0; n <= 8; n++)
        {
                for (int i = 0; i <= n; i++)
                {
                        cout << "Asset Price at Binomaial Tree Node("
                                << n << "," << i << ") = "
                                << fixed << setprecision(2);
                        cout << prices[index++];
                        cout << endl;
                }
        }
        return 0;
}
```

What we learned from BinomialTreeModel03.cpp:

- One-Dimensional Array:
  - const int SIZE = 81;
  - double prices[SIZE];
  - for (int i = 0; i < SIZE; i++)
    - prices[i] = 0.0;
- An array is a series of elements of the same type placed in **contiguous** memory locations that can be individually referenced by adding an index to a unique identifier.
  - What is Big-O for reading and writing one value from to an array?
  - What is the Big-O for adding and deleting a value from the front of the array?
- There is NO boundary check for an array.

BinomialTreeModel04.cpp

```
// Validate data before calculating asset price on the Binomial Tree
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
double u = 1.15125, d = 0.86862, r = 1.00545;
double s0 = 106.00;
if (s0 <= 0.0 || u <= 0.0 || d <= 0.0 || r <= 0.0 || u <= d)
{
        cerr << "Invalid data, terminate program without calculation" << endl;
        return -1;
}
```

```
    if (r >= u || r <= d)
    {
        cerr << "Arbitrage exists, terminate program without calculation" << endl;
        return -1;
    }
    // Compute asset price at every node on the Binomial Tree
    for (int n = 0; n <= 8; n++)
    {
        for (int i = 0; i <= n; i++)
        {
            cout << "Asset Price at Binomaial Tree Node("
            << n << "," << i << ") = " << fixed << setprecision(2);
            cout << s0 * pow(u, i) * pow(d, n - i) << endl;
        }
    }
    return 0;
}
```

What we learned from BinomialTreeModel04.cpp:

- Validate the data used for calculating asset price values on a Binomial Tree. If any the data used for calculation is invalid or there is an arbitrage, the program will be terminated without further calculation.
- Return nonzero when validation fails.
- Logic operator OR in the if statement.

BinomialTreeModel.h

```
#pragma once
namespace fre {
        //compute risk-neutral probability
        double RiskNeutProb(double U, double D, double R);

        //compute the asset price at node n,i
        double CalculateAssetPrice(double S0, double U, double D, int n, int i);

        //input, display, and check model data
        int GetInputData(double& S0, double& U, double& D, double& R);

        //validate input data for Binomial Tree Model
        int ValidateInputData(const double& S0, const double& U, const double& D,
        const double& R);
}
```

BinomialTreeModel.cpp

```
#include "BinomialTreeModel.h"
#include <iostream>
#include <cmath>
using namespace std;
namespace fre {
    //compute risk-neutral probability
    double RiskNeutProb(double U, double D, double R)
    {
        return (R - D) / (U - D);
    }
    //compute the asset price at node n,i
    double CalculateAssetPrice(double S0, double U, double D, int n, int i)
    {
        return S0 * pow(U, i) * pow(D, n - i);
    }
```

```
//input and display, check model data
  int GetInputData(double& S0, double& U, double& D, double& R)
  {
    //entering data
    cout << "Enter S0: "; cin >> S0;
    cout << "Enter U:  "; cin >> U;
    cout << "Enter D:  "; cin >> D;
    cout << "Enter R:  "; cin >> R;
    cout << endl;

    //making sure that S0>0, U>D>0, R>0
    if (S0 <= 0.0 || U <= 0.0 || D <= 0.0 || U <= D || R <= 0.0)
    {
      cout << "Illegal data ranges" << endl;
      cout << "Terminating program" << endl;
      return -1;
    }
    //checking for arbitrage
    if (R >= U || U <= D)
    {
        cout << "Arbitrage exists" << endl;
        cout << "Terminating program" << endl;
        return -1;
    }

    cout << "Input data checked" << endl;
    cout << "There is no arbitrage" << endl << endl;

    return 0;
}
```

```
int ValidateInputData(const double& S0, const double& U,
                                    const double& D, const double& R)
  {  //making sure that S0>0, U>D>0, R>0
     if (S0 <= 0.0 || U <= 0.0 || D <= 0.0 || U <= D || R <= 0.0)
     {
        cout << "Illegal data ranges" << endl;
        cout << "Terminating program" << endl;
        return -1;
     }
     //checking for arbitrage
     if (R >= U || U <= D)
     {  cout << "Arbitrage exists" << endl;
        cout << "Terminating program" << endl;
        return -1;
     }
     cout << "Input data checked" << endl;
     cout << "There is no arbitrage" << endl << endl;
     return 0;
  }
}
```

BinomialTreeModel05.cpp

```
#include "BinomialTreeModel.h"
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
using namespace fre;
int main()
{
    double u = 1.15125, d = 0.86862, r = 1.00545;
    double s0 = 106.00;

    if (ValidateInputData(s0, u, d, r) == -1)
    return -1;
    // Compute asset price at every node on the Binomial Tree
    for (int n = 0; n <= 8; n++)
    {
        for (int i = 0; i <= n; i++)
        {
            cout << "Asset Price at Binomaial Tree Node(" << n << "," << i << ") = "
            << fixed << setprecision(2);
            cout << CalculateAssetPrice(s0, u, d, n, i);
            cout << endl;
        }
    }
    return 0;
}
```

What we learned from BinomialTreeModel05.cpp:

- Function Declaration in user-defined header file:
  - **BinomialTreeModel.h**
- Function Definition in cpp file:
  - **BinomialTreeModel.cpp**
- Invoke Binomial Tree Model functions in main() function:
  - **BinomialTreeModel05.cpp**
  - **Call by Value (Passed by Value) and Call by Reference (Passed by Reference)**

- **Call by Value:**
  - passing arguments to a function copies the actual value of an argument into the parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.

- **Call by Reference:**
  - A reference variable is an alias, that is, another name for an already existing variable. Once a reference is initialized with a variable, either the variable name or the reference name may be used to refer to the same variable.
  - The parameter is an alias of the argument. In this case, changes made to the parameter inside the function have also change the argument.

```
#include <iostream>
using namespace std;
void Foo(int a, int& b)
{
        a++;
        b++;
        cout << "In Function Foo a = " << a << " and b = " << b << endl;
}
int main()
{
        int x = 1, y = 1;
        Foo(x, y);
        cout << "In main function x = " << x << " and y = " << y << endl;
        return 0;
}
/*
In function Foo a = 2 and b = 2
In main function x = 1 and y = 2
*/
```

Question?

- What happen if it is

```
void Foo(int a, const int& b)
{
        a++;
        b++;
        cout << "In Function Foo a = " << a
                << " and b = " << b << endl;
}
```

CRR Pricer

- Within the Binomial Tree Model, the price H(n,i) at each time step n and node i of a European option with expiry date N and payoff h(S(N)) can be computed using the Cox-Ross-Rubistein (CRR) procedure.

Option01.h

```
#pragma once

namespace fre {
//pricing European option
double PriceByCRR(double S0, double U, double D, double R,
                  int N, double K);

//computing call payoff
double CallPayoff(double z, double K);
}
```

Option01.cpp

```
#include "Option01.h"
#include "BinomialTreeModel.h"
#include <iostream>
#include <cmath>
using namespace std;
namespace fre {
double PriceByCRR(double S0, double U, double D, double R, int N, double K)
{
        double q = RiskNeutProb(U, D, R);
        double Price[N+1];
        for (int i = 0; i < sizeof(Price)/sizeof(Price[0]); i++)
            Price[i] = 0.0;
        for (int i = 0; i <= N; i++)
        {
            Price[i] = CallPayoff(CalculateAssetPrice(S0, U, D, N, i), K);
        }
        for (int n = N - 1; n >= 0; n--)
        {
            for (int i = 0; i <= n; i++)
            {
                Price[i] = (q * Price[i + 1] + (1 - q) * Price[i]) / R;
            }
        }
        return Price[0];
    }
    double CallPayoff(double z, double K)
    {
        if (z > K) return z - K;
        return 0.0;
    }
}
```

OptionPricer01.cpp

```
#include "BinomialTreeModel.h"
#include "Option01.h"
#include <iostream>
#include <iomanip>
using namespace std;
using namespace fre;
int main()
{   double u = 1.15125, d = 0.86862, r = 1.00545;
    double s0 = 106.00, k = 100.00;
    const int N = 8;
    double optionPrice = PriceByCRR(s0, u, d, r, N, k);
    cout << "European call option price = " << fixed <<
           setprecision(2) << optionPrice << endl;
    return 0;
}
// European call option price = 21.68
```

Homework Assignment

- Modify the PriceByCRR() function in Option01.cpp to compute the current price (time 0) of a European option using CRR formula:

$$H(0) = \frac{1}{(R)^N} \sum_{i=0}^{N} \frac{N!}{i!\,(N-i)!} q^i (1-q)^{N-i} h(S(N,i))$$

References

- Numerical Methods in Finance with C++ (Mastering Mathematical Finance), by Maciej J. Capinski and Tomasz Zastawniak, Cambridge University Press, 2012, ISBN-10: 0521177162