# BS1029: Computational Finance with C++

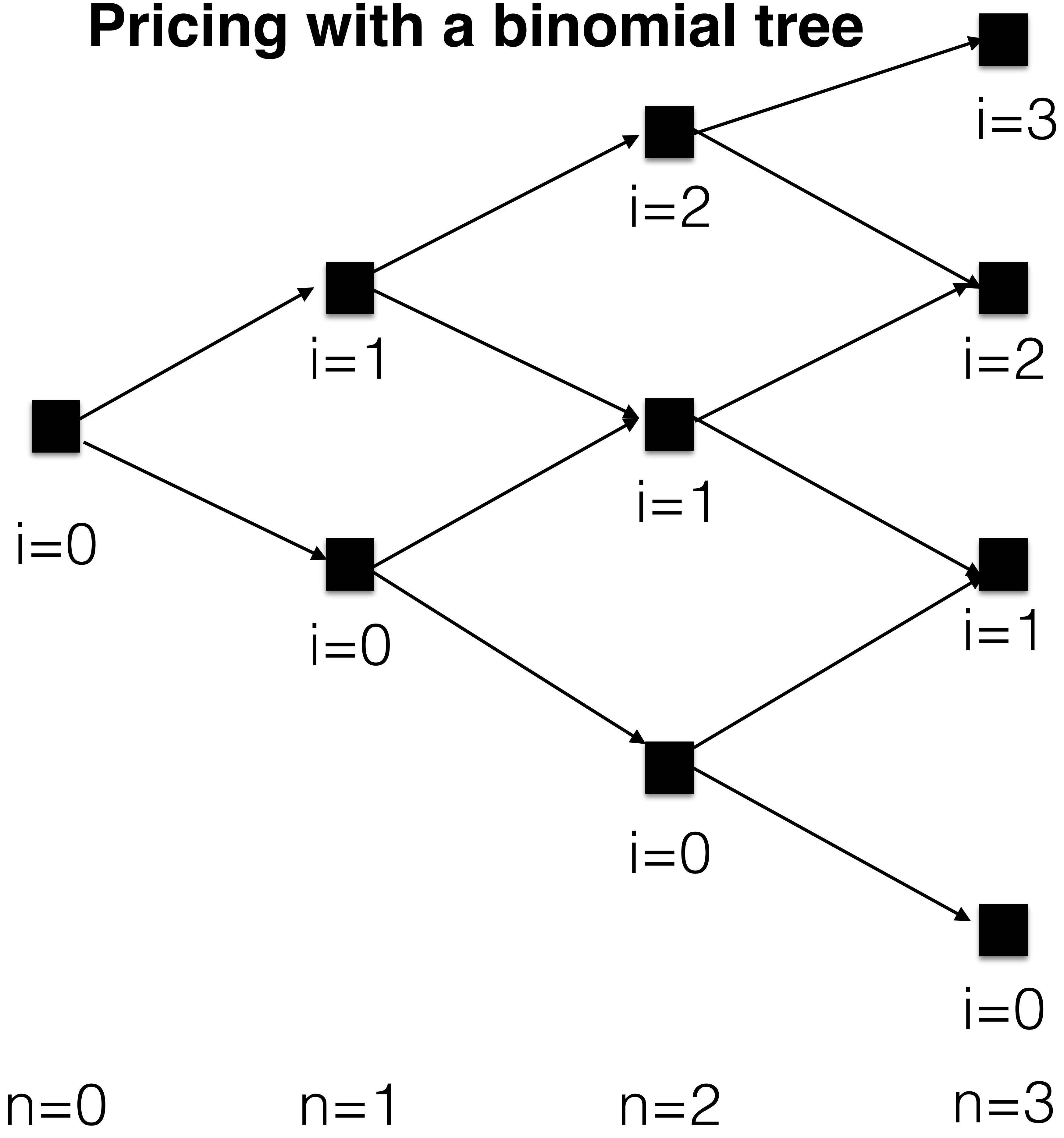
## Lecture 1: Elements of procedural programming

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

# Outline

- Binomial Pricing: Cox-Ross-Rubenstein Model.
- Reminder: Function Calling
    - ✦Function calls by value
    - ✦Function calls by reference
- Compilation with several files
- Reminder: Pointers
- Function Pointers
- Reading:
    - ✦Chapter 1 Capinski+Zastawniak,Numerical Methods in finance with C++

## Pricing with a binomial tree



**U:** up factor >-1
**D:** down factor >-1
**S(n,i):** Asset Price
At time n, state i
$S(0)(1+U)^i(1+D)^{n-i}$

**S(0)>0**
**R:** risk free rate
**No Arbitrage:**
D<R<U

# Cox-Ros-Rubenstein Procedure

- H(n,i)=price of a European option at **time** n, **node** i
- At expiry time=N and payoff is h(S(N)) e.g. h(S(N))=max(S(N)-K,0) for European call option with strike K.
- **CRR procedure** work by backward induction: H(n+1,i) known and then:

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

**Risk Neutral Probabilities:** $q = \dfrac{R-D}{U-D}$

# Entering Data:`Main02.cpp`

**Things to note:**

- include `cmath,iostream`
- `using namespace std`
- `cout<< vs cin>>`
- if statements and error checking

Exercise 1.1: Modify the code so that the user can enter n and i from the keyboard

## Functions: `Main03.cpp`

**Things to note:**

- Separate functions for different tasks
- Order of functions in file is important!
- Pass by value.
- Pass by reference.
- Logical equality operator vs assignment

**Exercise 1.2:** Write a function called interchange that interchanges the contents of two variables of type double which are to be passed to the function by reference.

# Separate Compilation: `BinModel01.cpp, Main04.cpp`

**Things to note:**

- Using several files
- Function prototypes

# CRR Pricer: `Options01.cpp`, `Main05.cpp`

**Things to note:**

- Function overloading
- Arrays
- Loops
- Order of function declarations in options01.h

## Exercises

**Exercise 1.4:** Include checking for input data integrity in the GetInputData() function in Options01.cpp. You want to ensure that $0<K$ and $0<N$.

**Exercise 1.5:** Modify the PriceByCRR() function in Options01.cpp to compute the time 0 price of a European option using the Cox-Ros-Rubinstein (CRR) formula

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

instead of the iterative process in Slide 4

# Pointers: `Options02.cpp`, `Main06.cpp`

**Things to note:**

- Pointers
- Arrays and pointers

**Exercise 1.8:** Modify the function GetInputData() from BinModel01.h and BinModel01.cpp to have the parameters passed to it by pointers rather than by reference. What changes are needed in Main04.cpp to call the modified function?

# Function Pointers: `Options03.cpp, Main07.cpp`

**Things to note:**

- Function pointers
- Why and when to use them?

**Exercise 1.9:** The payoff of a digital call with strike price K is

$$h^{digit\ call}(z) = \begin{cases} 1 \text{ if } K < z \\ 0 \text{ otherwise} \end{cases}$$

Include the ability to price digital calls in the program developed so far by adding the new payoff function to the file Options03.cpp, just as was done for calls and puts, and suitably modifying the code in Options03.h and Main07.cpp to compute the prices of a digital call option.

# Function Pointers: `Options03.cpp, Main07.cpp`

**Exercise 1.10:** The payoff of a digital put with strike price K is

$$h^{digit\ put}(z) = \begin{cases} 1 \text{ if } K > z \\ 0 \text{ otherwise} \end{cases}$$

Add the ability to price digital puts to the program developed above by placing any new code in a separate .cpp file, complete with its own header file, without altering anything at all in the original files, except for the appropriate changes in Main07.cpp

# Summary

Used mainly **procedural** programming techniques:

- Pointers, function pointers
- Function overloading
- Function prototypes

We will look at object **oriented programming** techniques next

- Path dependent options
- American options
- Compute hedging portfolio
- Black and Scholes formula
- Monte Carlo Pricer

**Exercise 1.11:** Modify the function pointer Payoff in Options03.cpp so that the function pointed to will accept an array of type double in place of the single variable of type double that was used to pass the strike price.

Modify the payoff functions CallPayOff() and PutPayoff() for calls and puts to use with the new function pointer, and modify any remaining code as necessary to work with these.

Take advantage of the new functionality to price double digital options.

$$h^{dc}(z) = \begin{cases} 1 & \text{if } K_1 < z < K_2 \\ 0 & \text{otherwise} \end{cases}$$