FRE6883
Financial Computation

What are prerequisites?

- Have knowledge about derivates and quantitative methods of option pricing.
- Be aware of elementary programming concepts such as variable declaration and flow control structures such as sequential, looping and conditional statements in general.
- Knowledge in another programming language such as Python is helpful. But it is important not try to fit C/C++ into the language you already know.

Why C++?

- C++ is an industry standard because of its flexibility and performance. It has been the selected language for backend engine in many software applications. Therefore, C++ is a powerful modern object-oriented language.
- Many 3rd party libraries have been created to integrate market data and conduct complex numerical calculations in C++ applications with high efficiency.

The Benefit of Learning C++

- C++ is very "close to the hardware". This means that memory could be allocated and deallocated in a custom-defined manner, and hardware components such as CPU and network interface card could be manipulated directly via system call in C++ applications.
- C++ is extremely versatile. A business question could be programmed in many ways.
- C++ is an ISO standard and will be around for long time.

The Difficulty of Learning C++

- C++ may not be an ideal language for beginners. It provides far more flexibility to programmers often than required and could confusing.
- It is very easy to make mistakes and have bugs especially when pointers are used.
- The object-oriented program concept is not easy to grasp.
- The good news is that once you know C++, learning other programming languages become easier.

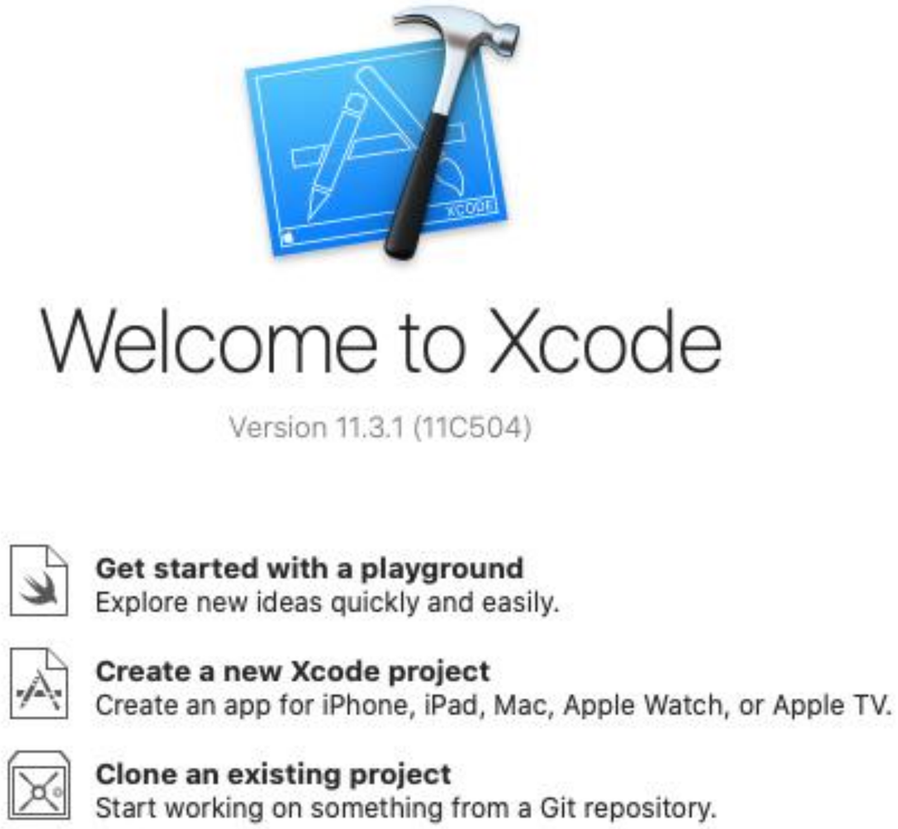
The Different Styles in C++ Programming

- Structured/Procedural Programming, C-Style: using functions and function calls, not object-oriented.
- Object-Oriented Programming: Encapsulating data and associated functions in classes, inheritance and polymorphism.
- Generic Programming and Templates: Type-independent classes for code reusability.

C/C++ Integrated Development Environment

- MAC: XCode, should be preinstalled, otherwise, download from Apple Web site.
- PC: Microsoft Visual Studio Community 2019, a free version of Microsoft Visual Studio.

C/C++ IDE for MAC



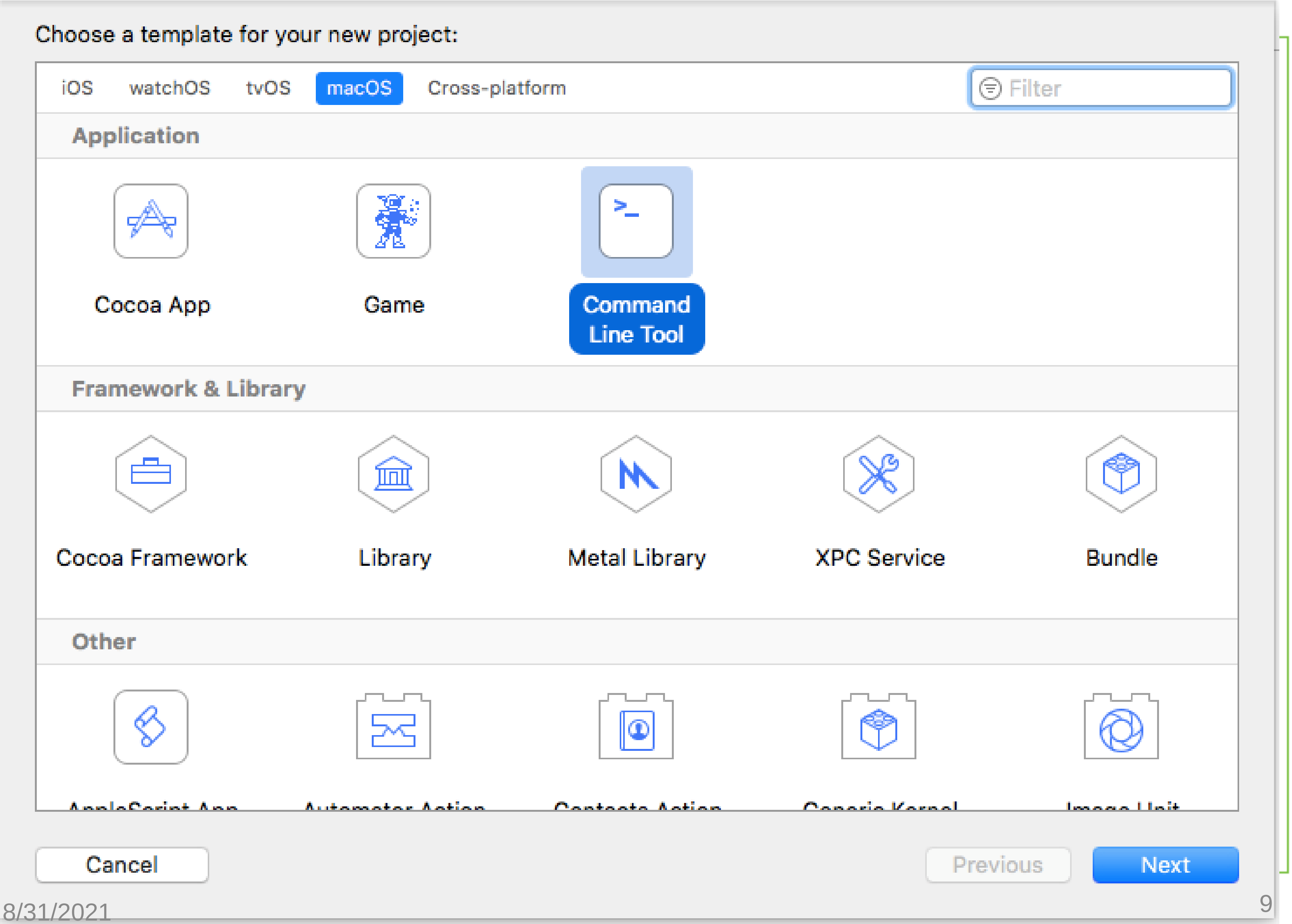
Welcome to Xcode

Version 11.3.1 (11C504)

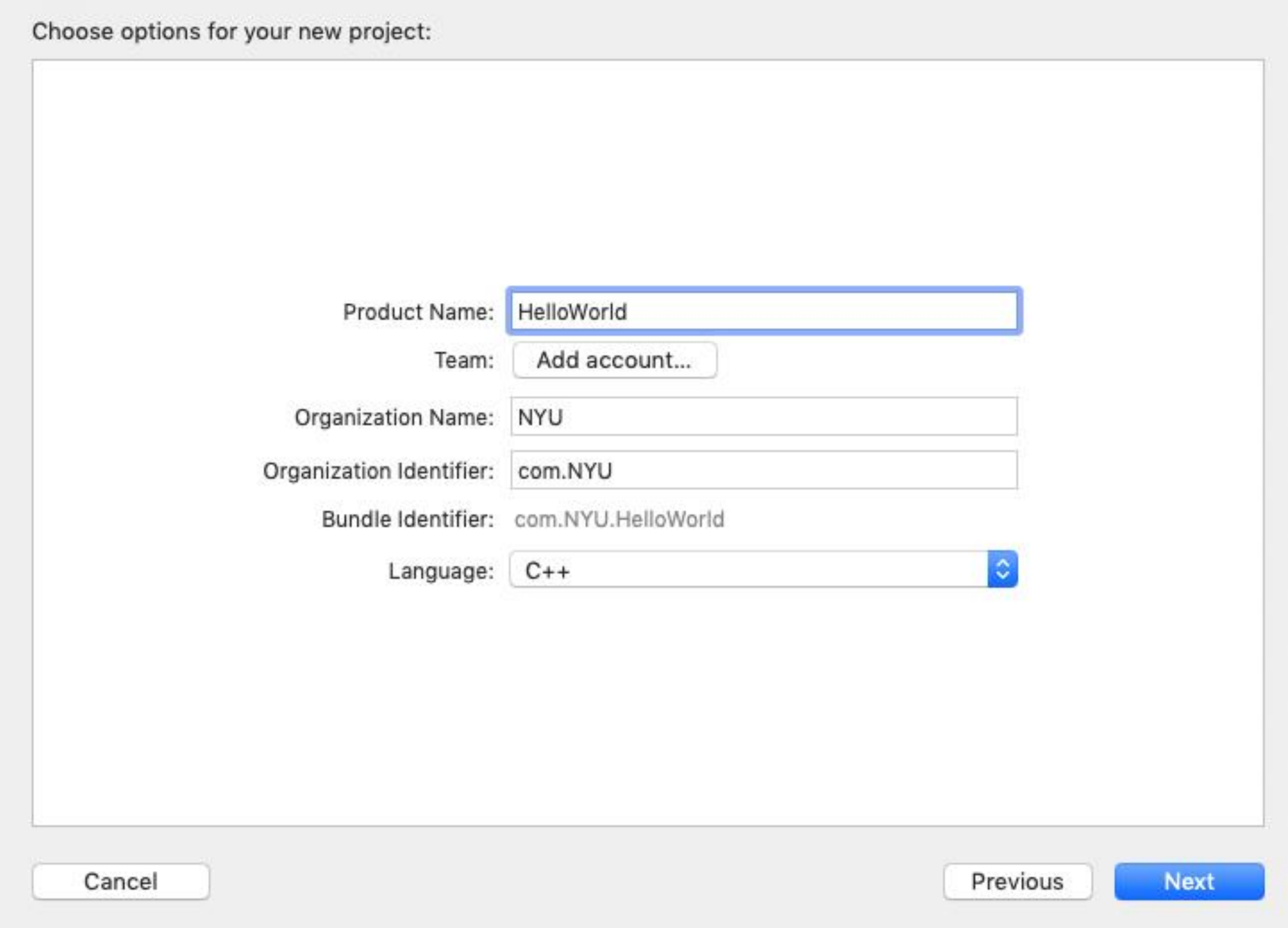
Get started with a playground
Explore new ideas quickly and easily.

Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.

Clone an existing project
Start working on something from a Git repository.



Choose a template for your new project:



Choose options for your new project:

Product Name: HelloWorld

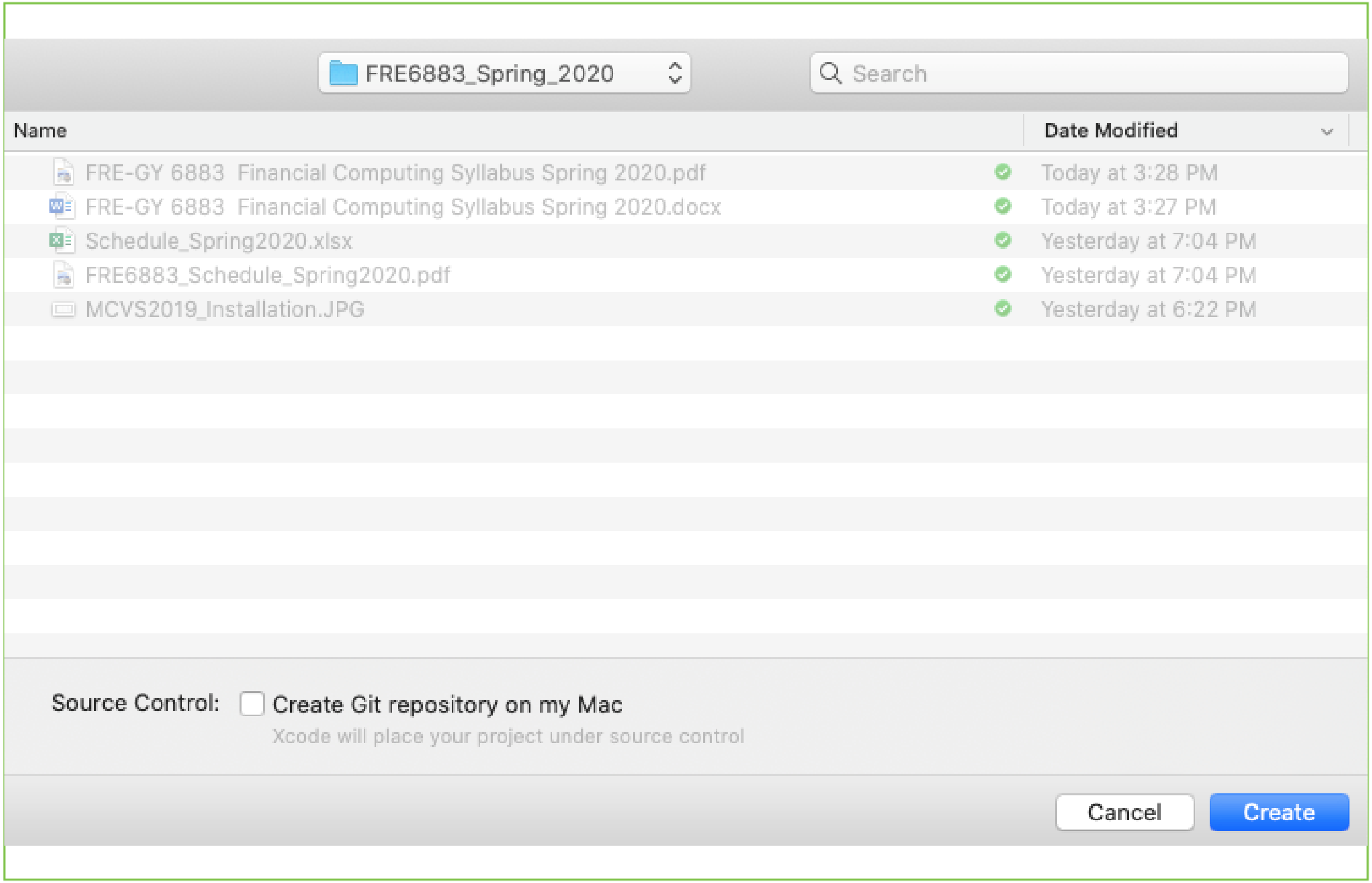
Team: Add account...

Organization Name: NYU

Organization Identifier: com.NYU

Bundle Identifier: com.NYU.HelloWorld

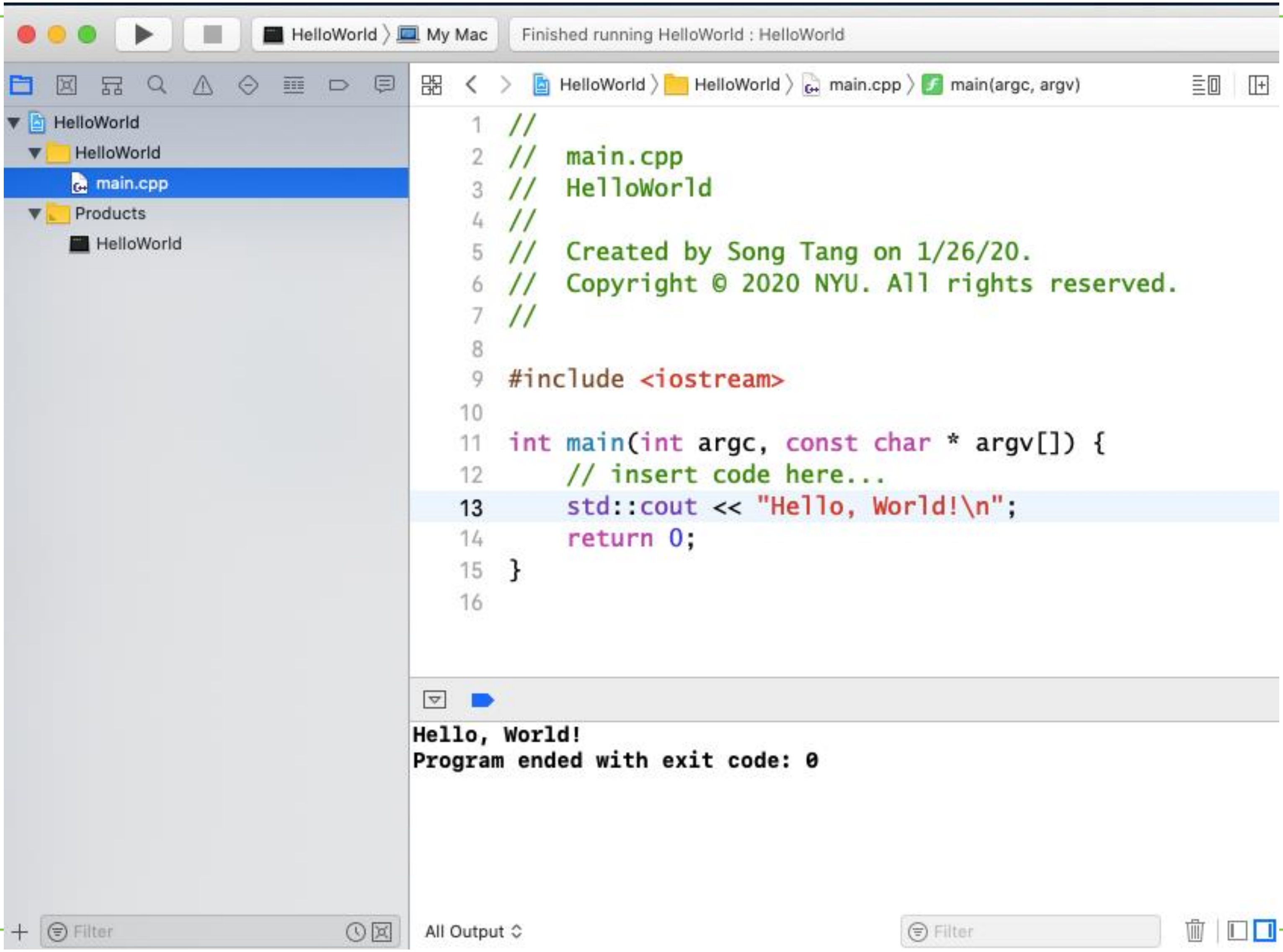
Language: C++



Name	Date Modified
FRE-GY 6883 Financial Computing Syllabus Spring 2020.pdf ✓	Today at 3:28 PM
FRE-GY 6883 Financial Computing Syllabus Spring 2020.docx ✓	Today at 3:27 PM
Schedule_Spring2020.xlsx ✓	Yesterday at 7:04 PM
FRE6883_Schedule_Spring2020.pdf ✓	Yesterday at 7:04 PM
MCVS2019_Installation.JPG ✓	Yesterday at 6:22 PM

Source Control: □ Create Git repository on my Mac

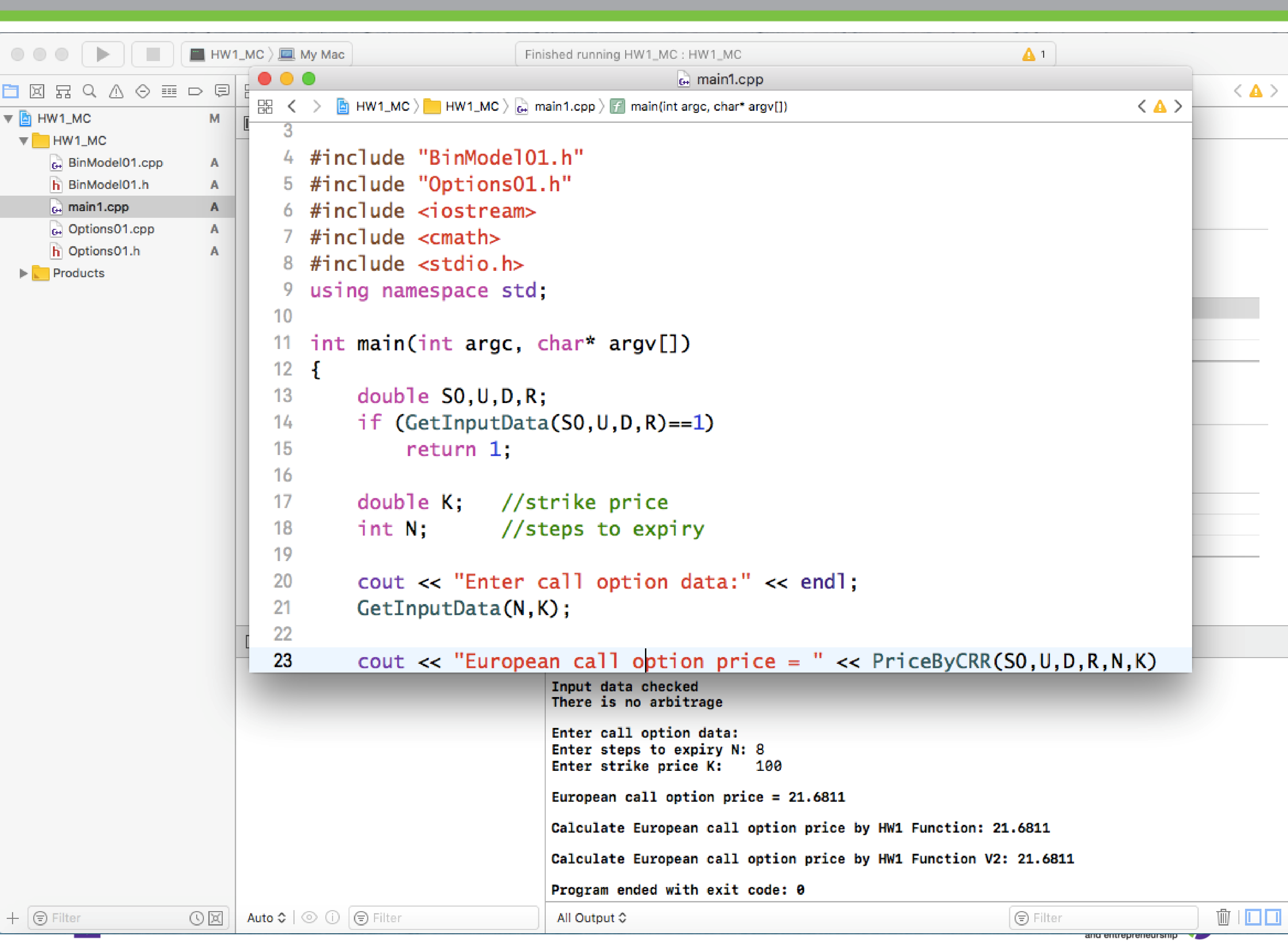
Xcode will place your project under source control



HelloWorld 〉 My Mac
Finished running HelloWorld : HelloWorld
HelloWorld 〉 HelloWorld 〉 main.cpp 〉 main(argc, argv)
HelloWorld
HelloWorld
main.cpp
Products
HelloWorld
Filter
All Output ◇
Filter

```
//
//  main.cpp
//  HelloWorld
//
//  Created by Song Tang on 1/26/20.
//  Copyright © 2020 NYU. All rights reserved.
//

#include <iostream>

int main(int argc, const char * argv[]) {
    // insert code here...
    std::cout << "Hello, World!\n";
    return 0;
}

```

```
Hello, World!
Program ended with exit code: 0
```

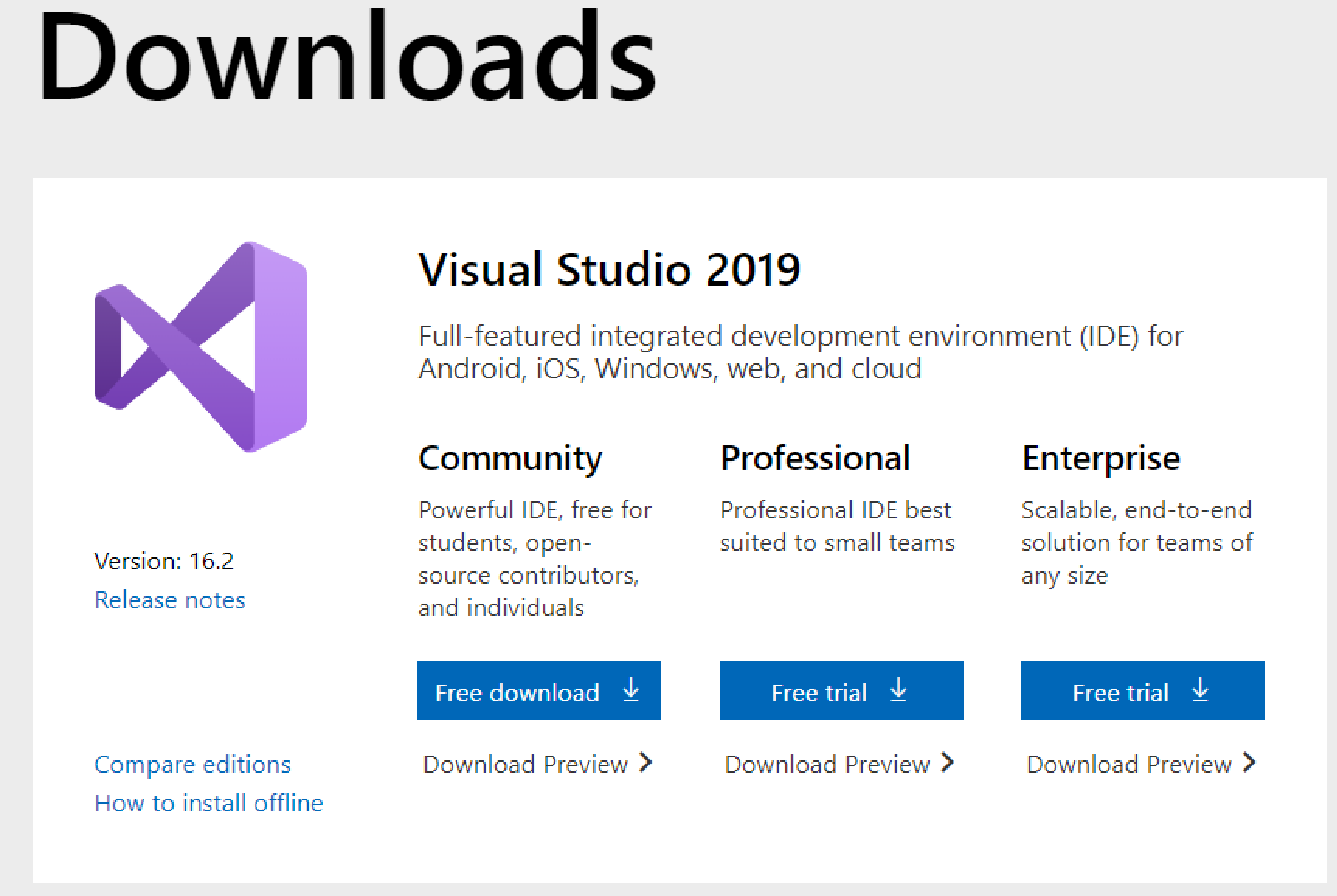


```cpp

#include "BinModel01.h"
#include "Options01.h"
#include <iostream>
#include <cmath>
#include <stdio.h>
using namespace std;

int main(int argc, char* argv[])
{
    double S0,U,D,R;
    if (GetInputData(S0,U,D,R)==1)
        return 1;

    double K;   //strike price
    int N;      //steps to expiry

    cout << "Enter call option data:" << endl;
    GetInputData(N,K);

    cout << "European call option price = " << PriceByCRR(S0,U,D,R,N,K)
```

```
Input data checked
There is no arbitrage

Enter call option data:
Enter steps to expiry N: 8
Enter strike price K:    100

European call option price = 21.6811

Calculate European call option price by HW1 Function: 21.6811

Calculate European call option price by HW1 Function V2: 21.6811

Program ended with exit code: 0
```

C/C++ Development Environment for PC

- Download Microsoft Visual Studio 2019 **Community Version**
- https://visualstudio.microsoft.com/downloads/



Downloads

Visual Studio 2019

Full-featured integrated development environment (IDE) for Android, iOS, Windows, web, and cloud

Version: 16.2

Release notes

Community

Powerful IDE, free for students, open-source contributors, and individuals

Free download ⤓

Download Preview ›

Professional

Professional IDE best suited to small teams

Free trial ⤓

Download Preview ›

Enterprise

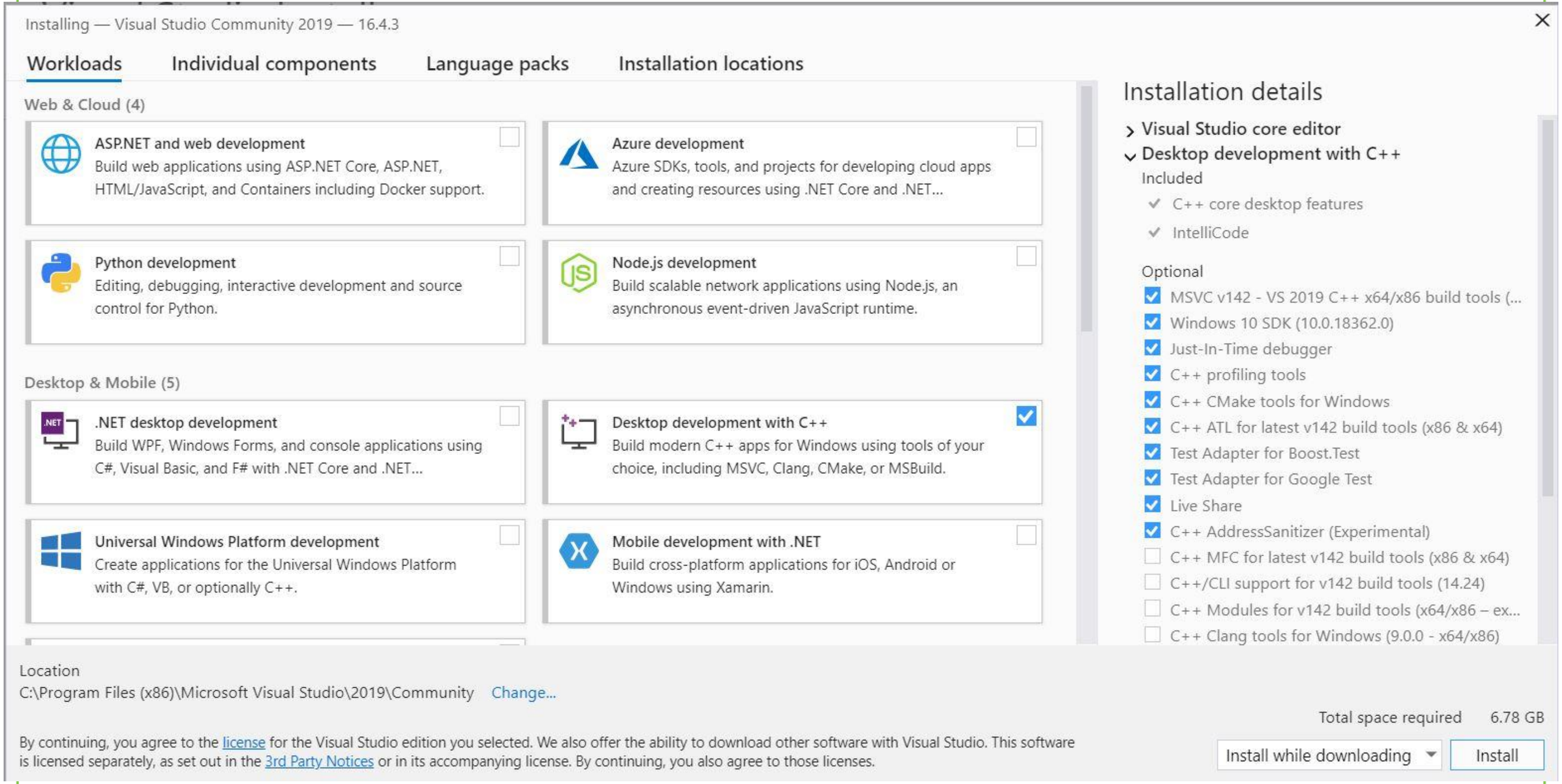
Scalable, end-to-end solution for teams of any size

Free trial ⤓

Download Preview ›

Compare editions

How to install offline



Installing — Visual Studio Community 2019 — 16.4.3

Web & Cloud (4)

- □ ASP.NET and web development
  Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.
- □ Azure development
  Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET...
- □ Python development
  Editing, debugging, interactive development and source control for Python.
- □ Node.js development
  Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- □ .NET desktop development
  Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET...
- ☑ Desktop development with C++
  Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- □ Universal Windows Platform development
  Create applications for the Universal Windows Platform with C#, VB, or optionally C++.
- □ Mobile development with .NET
  Build cross-platform applications for iOS, Android or Windows using Xamarin.

Installation details

> Visual Studio core editor

⌄ Desktop development with C++

Included

- ✓ C++ core desktop features
- ✓ IntelliCode

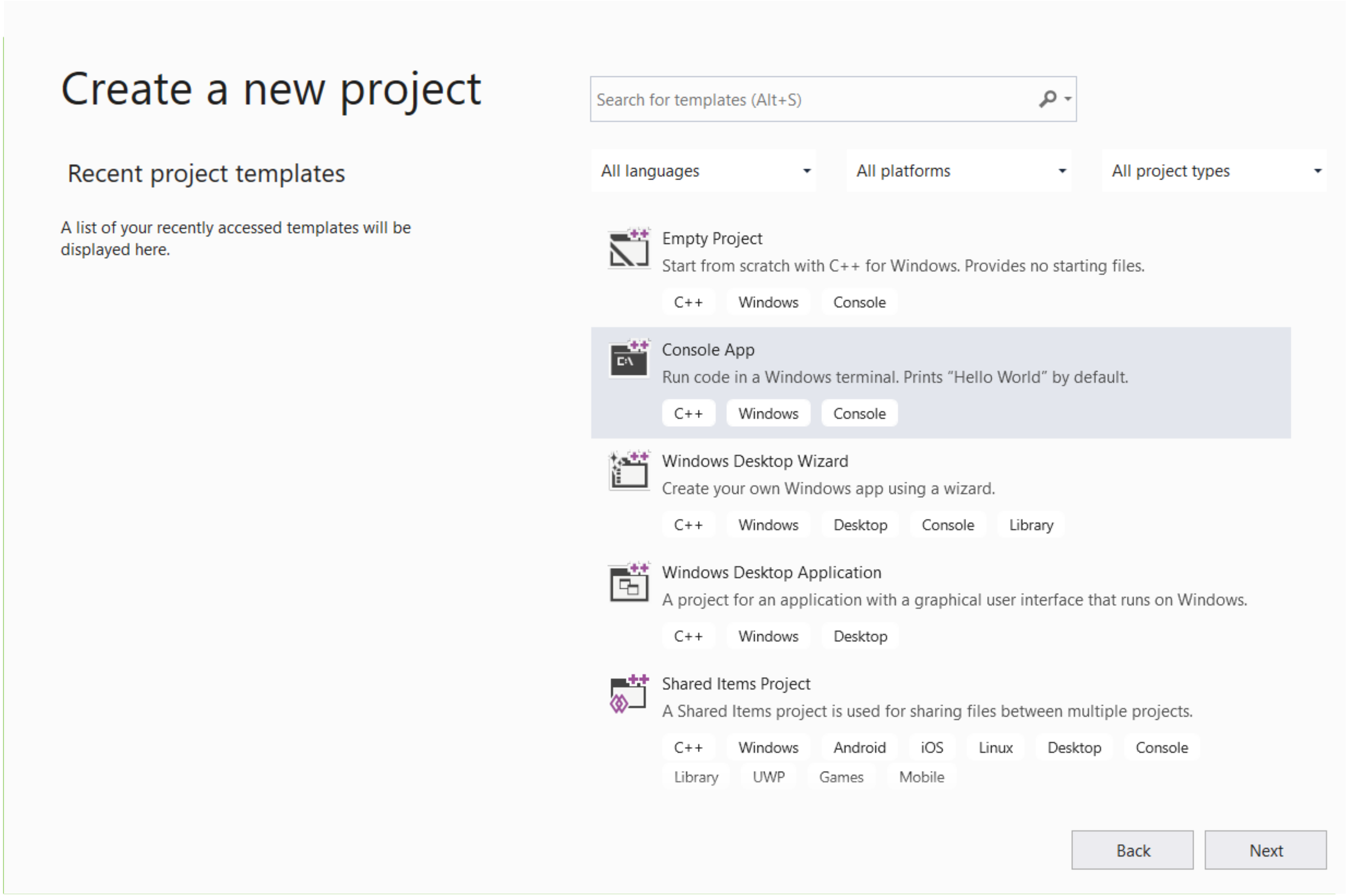
Optional

- ☑ MSVC v142 - VS 2019 C++ x64/x86 build tools (...
- ☑ Windows 10 SDK (10.0.18362.0)
- ☑ Just-In-Time debugger
- ☑ C++ profiling tools
- ☑ C++ CMake tools for Windows
- ☑ C++ ATL for latest v142 build tools (x86 & x64)
- ☑ Test Adapter for Boost.Test
- ☑ Test Adapter for Google Test
- ☑ Live Share
- ☑ C++ AddressSanitizer (Experimental)
- □ C++ MFC for latest v142 build tools (x86 & x64)
- □ C++/CLI support for v142 build tools (14.24)
- □ C++ Modules for v142 build tools (x64/x86 – ex...
- □ C++ Clang tools for Windows (9.0.0 - x64/x86)

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community Change...

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 6.78 GB



Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

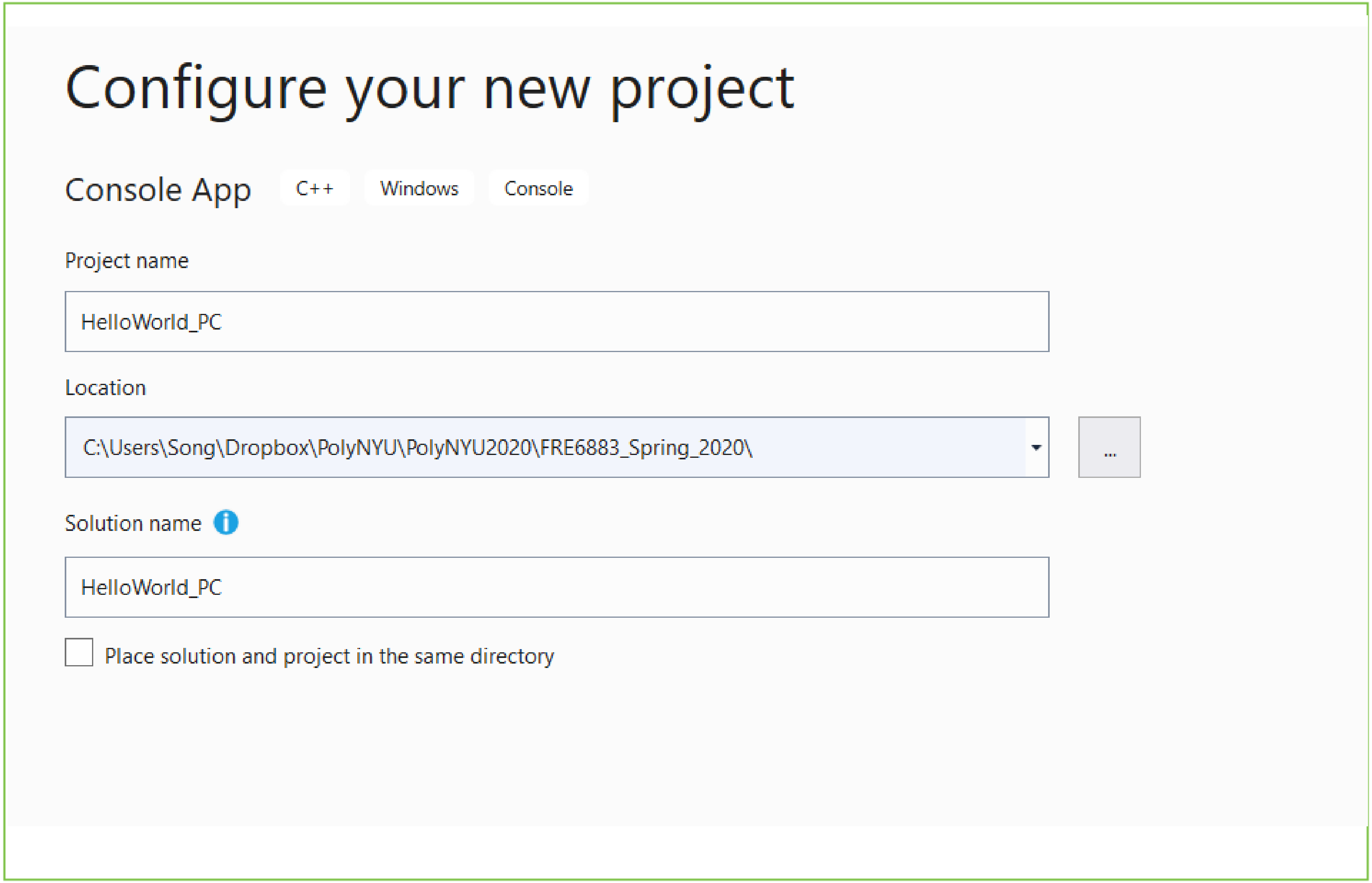
Empty Project
Start from scratch with C++ for Windows. Provides no starting files.
C++ Windows Console

Console App
Run code in a Windows terminal. Prints “Hello World” by default.
C++ Windows Console

Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ Windows Desktop Console Library

Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.
C++ Windows Desktop

Shared Items Project
A Shared Items project is used for sharing files between multiple projects.
C++ Windows Android iOS Linux Desktop Console Library UWP Games Mobile



Configure your new project

Console App

Project name

HelloWorld_PC

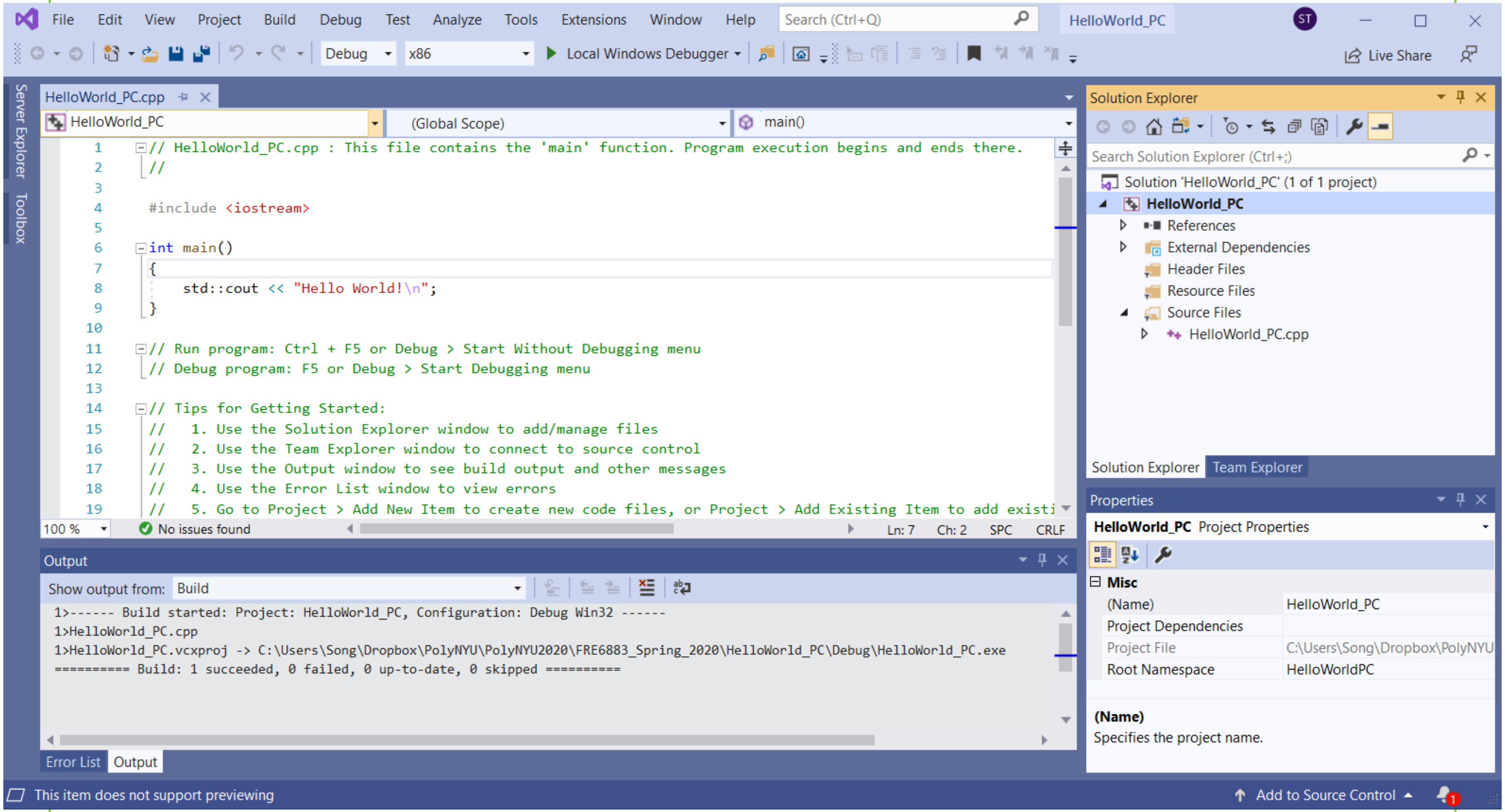
Location

C:\Users\Song\Dropbox\PolyNYU\PolyNYU2020\FRE6883_Spring_2020\

Solution name

HelloWorld_PC

□ Place solution and project in the same directory



```
// HelloWorld_PC.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}

// Run program: Ctrl + F5 or Debug > Start Without Debugging menu
// Debug program: F5 or Debug > Start Debugging menu

// Tips for Getting Started:
//   1. Use the Solution Explorer window to add/manage files
//   2. Use the Team Explorer window to connect to source control
//   3. Use the Output window to see build output and other messages
//   4. Use the Error List window to view errors
//   5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existi
```

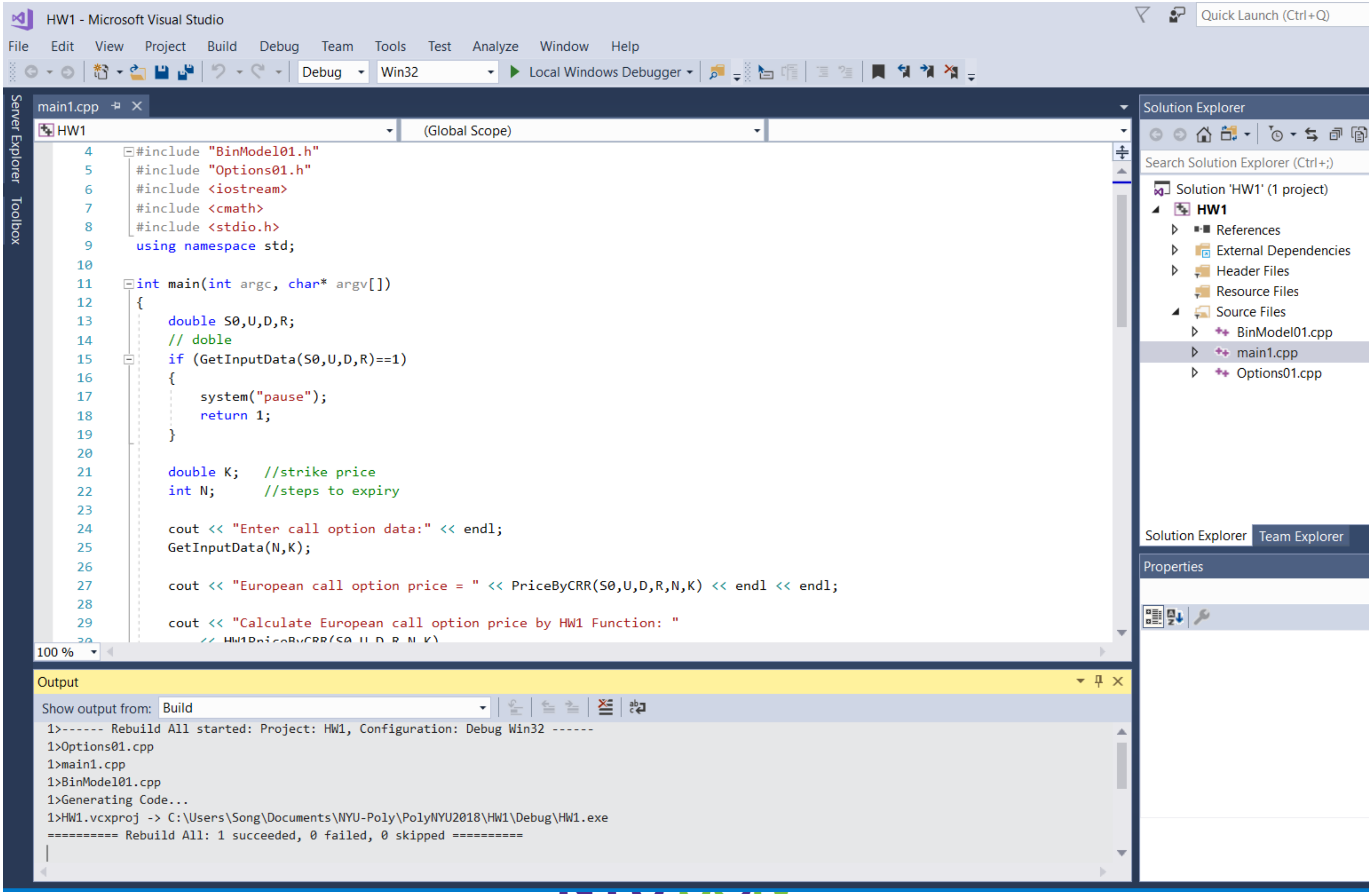
```
1>------ Build started: Project: HelloWorld_PC, Configuration: Debug Win32 ------
1>HelloWorld_PC.cpp
1>HelloWorld_PC.vcxproj -> C:\Users\Song\Dropbox\PolyNYU\PolyNYU2020\FRE6883_Spring_2020\HelloWorld_PC\Debug\HelloWorld_PC.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

HelloWorld_PC Project Properties

Misc

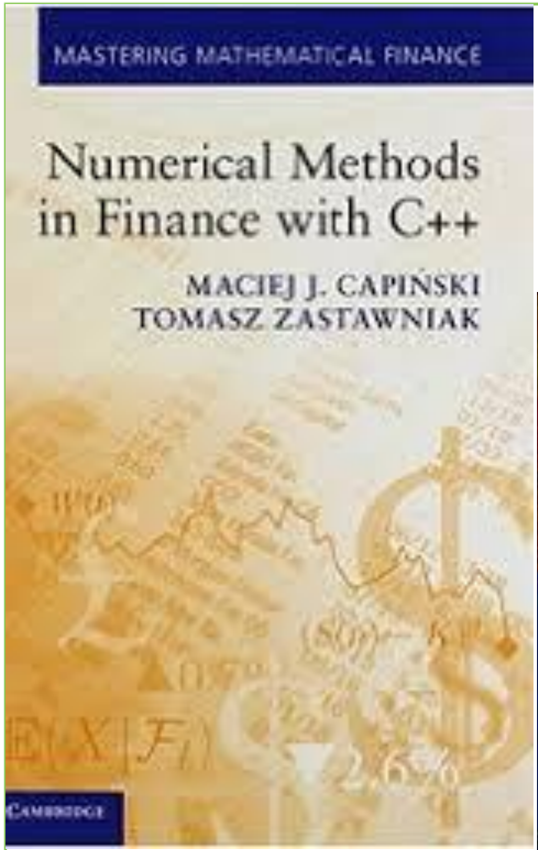
(Name)	HelloWorld_PC
Project Dependencies	
Project File	C:\Users\Song\Dropbox\PolyNYU
Root Namespace	HelloWorldPC

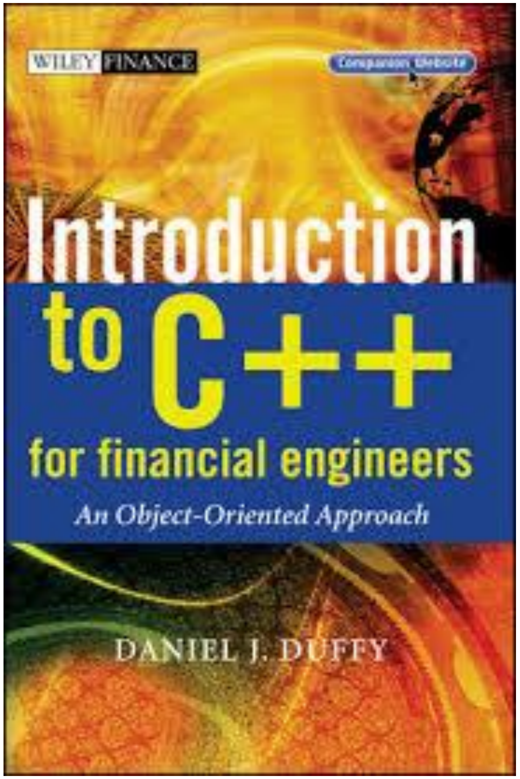
(Name)
Specifies the project name.

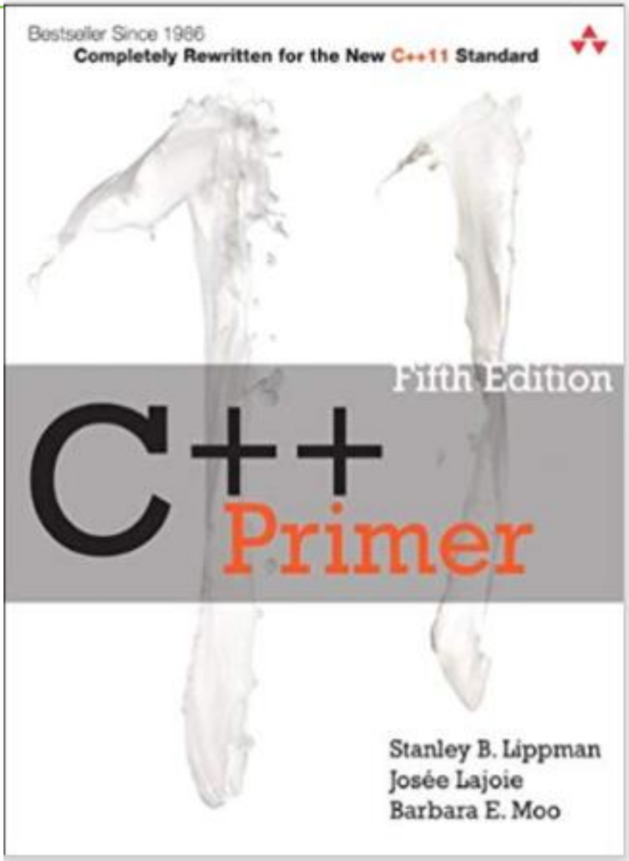


```
#include "BinModel01.h"
#include "Options01.h"
#include <iostream>
#include <cmath>
#include <stdio.h>
using namespace std;

int main(int argc, char* argv[])
{
    double S0,U,D,R;
    // doble
    if (GetInputData(S0,U,D,R)==1)
    {
        system("pause");
        return 1;
    }

    double K;   //strike price
    int N;      //steps to expiry

    cout << "Enter call option data:" << endl;
    GetInputData(N,K);

    cout << "European call option price = " << PriceByCRR(S0,U,D,R,N,K) << endl << endl;

    cout << "Calculate European call option price by HW1 Function: "
        << HW1PriceByCRR(S0,U,D,R,N,K)
```

```
1>------ Rebuild All started: Project: HW1, Configuration: Debug Win32 ------
1>Options01.cpp
1>main1.cpp
1>BinModel01.cpp
1>Generating Code...
1>HW1.vcxproj -> C:\Users\Song\Documents\NYU-Poly\PolyNYU2018\HW1\Debug\HW1.exe
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```


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

Topic 1:
Structured Programming Binomial Tree Model Implementation

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

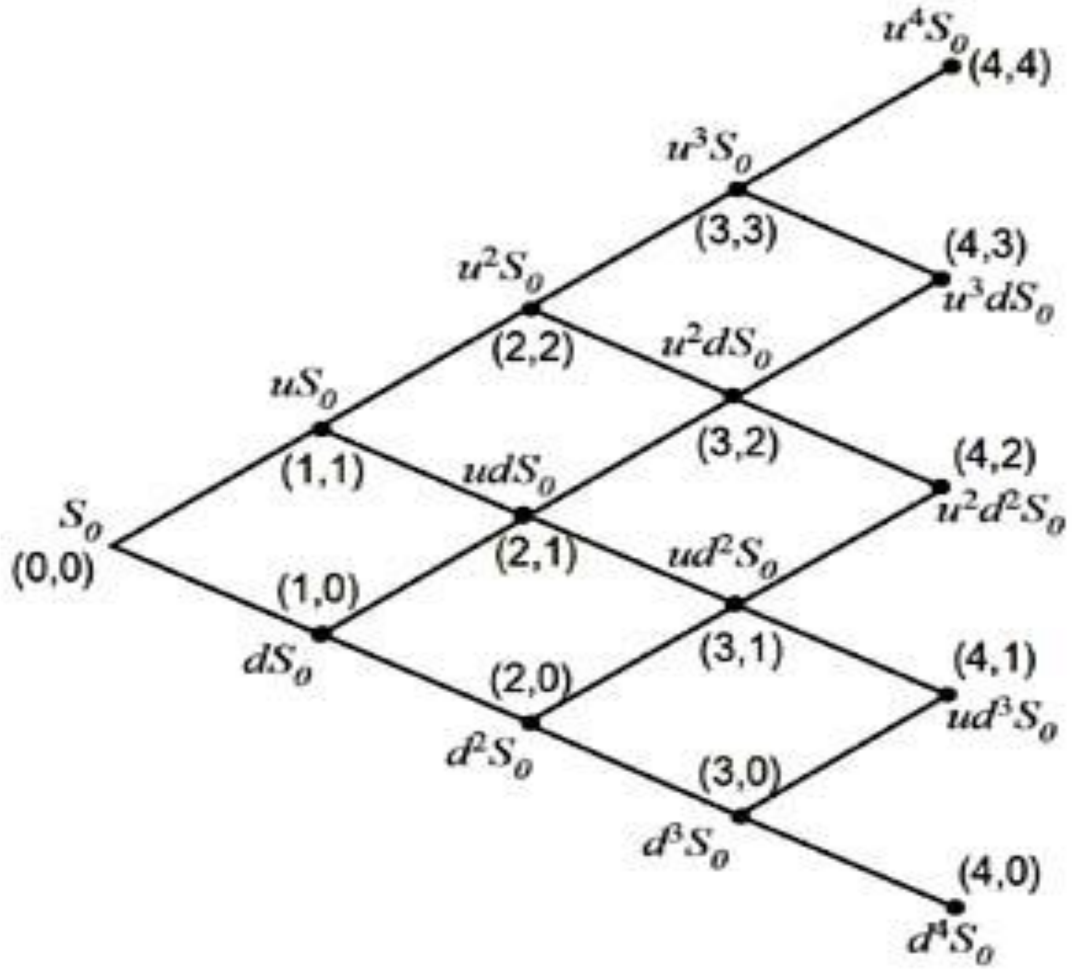


Figure 1. Binomial Tree Structure

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

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

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

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

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

```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);
         // If you are using MAX XCode
        // double Price[N];
        // If you use Microsoft Visual Studio 2019,
        double Price[9];
        for (int i = 0; i < sizeof(N); 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;
    }
}
```

OptionPricer02.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