CPLEX Concert Technology Tutorial

Tutorial By:
Dwarkanath Prabhu
Dr. Jorge Leon

June 2017

Table of Contents

1. About CPLEX and Concert

CPLEX is a suite of tools by ILOG (now under IBM) that solve linear programming, integer programming and mixed integer programming problems. There are several ways in which an LP or MIP can be solved using CPLEX. One of the most powerful ways is to define the LP in a programming language, using CPLEX to solve it and returning the results using the programming language in a user-friendly format. To do this, CPLEX provides libraries that can be called by several programming languages such as C++, Java, Python etc. 'Concert' is the name of the technology that provides an interface to call CPLEX libraries using C++, C# and JAVA. This tutorial uses C++.

2. Installing CPLEX and Visual Studio

Visual Studio is an Integrated Development Environment (IDE) i.e. a tool to build and maintain all code related to your projects. The CPLEX documentation recommends using Visual Studio to write C++ code. The "Community" Version is free for students and open-source developers. The Professional edition is for professional developers and universities. Texas A&M University provides access to Visual Studio Professional in some labs. If it is not available on the system or you wish to run the code on your personal computer, all versions can be downloaded from: https://www.visualstudio.com/.
CPLEX is free for students and academics. If it is not already installed on the system you are using (university or personal), it can be downloaded from:

a. For students: https://ibm.onthehub.com/WebStore/OfferingDetails.aspx?o=9b4eadea-9776-e611-9421-b8ca3a5db7a1
b. For academics: https://ibm.onthehub.com/WebStore/OfferingDetails.aspx?o=6fcc1096-7169-e611-9420-b8ca3a5db7a1

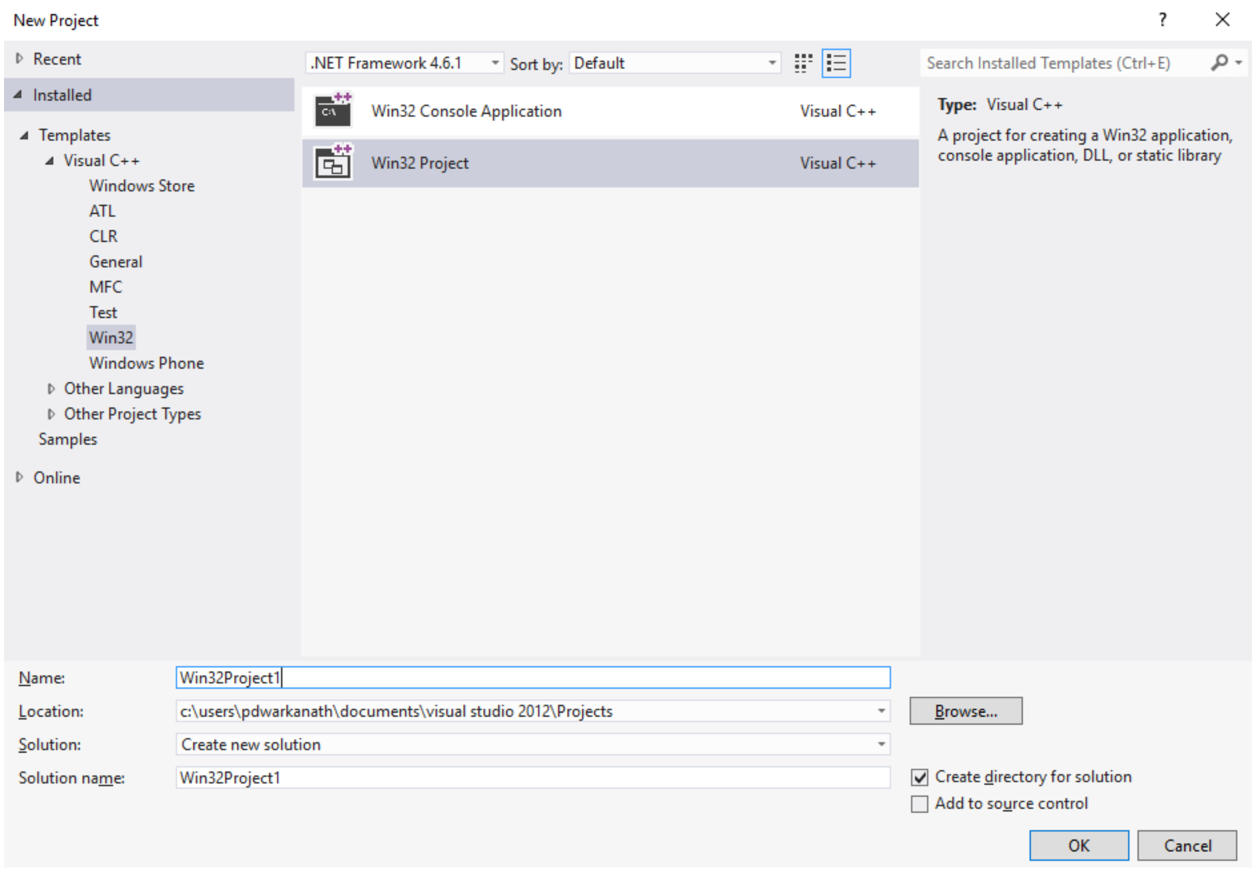
In the following steps, the folder in which CPLEX is installed on your computer will be referred to as `<CPLEXDIR>`. If using Windows and following defaults, this will be `C:\Program Files\IBM\ILOG\CPLEX_Studioxxxx` (Here xxxx is the version of CPLEX installed)
Please keep a note of the `<CPLEXDIR>` for future reference.

3. Setting up CPLEX Concert with C++ in Release Mode

Follow the steps in this order

a. Open a new project.

i. Open Visual Studio
ii. Select File > New > Project
iii. In the left pane, select Installed > Templates > Visual C++
iv. Select "Win32 Application"
v. Change name and location of project, if necessary.
vi. Click "OK"



New Project

Win32 Console Application	Visual C++
Win32 Project	Visual C++

Type: Visual C++

A project for creating a Win32 application, console application, DLL, or static library

Name: Win32Project1

Location: c:\users\pdwarkanath\documents\visual studio 2012\Projects

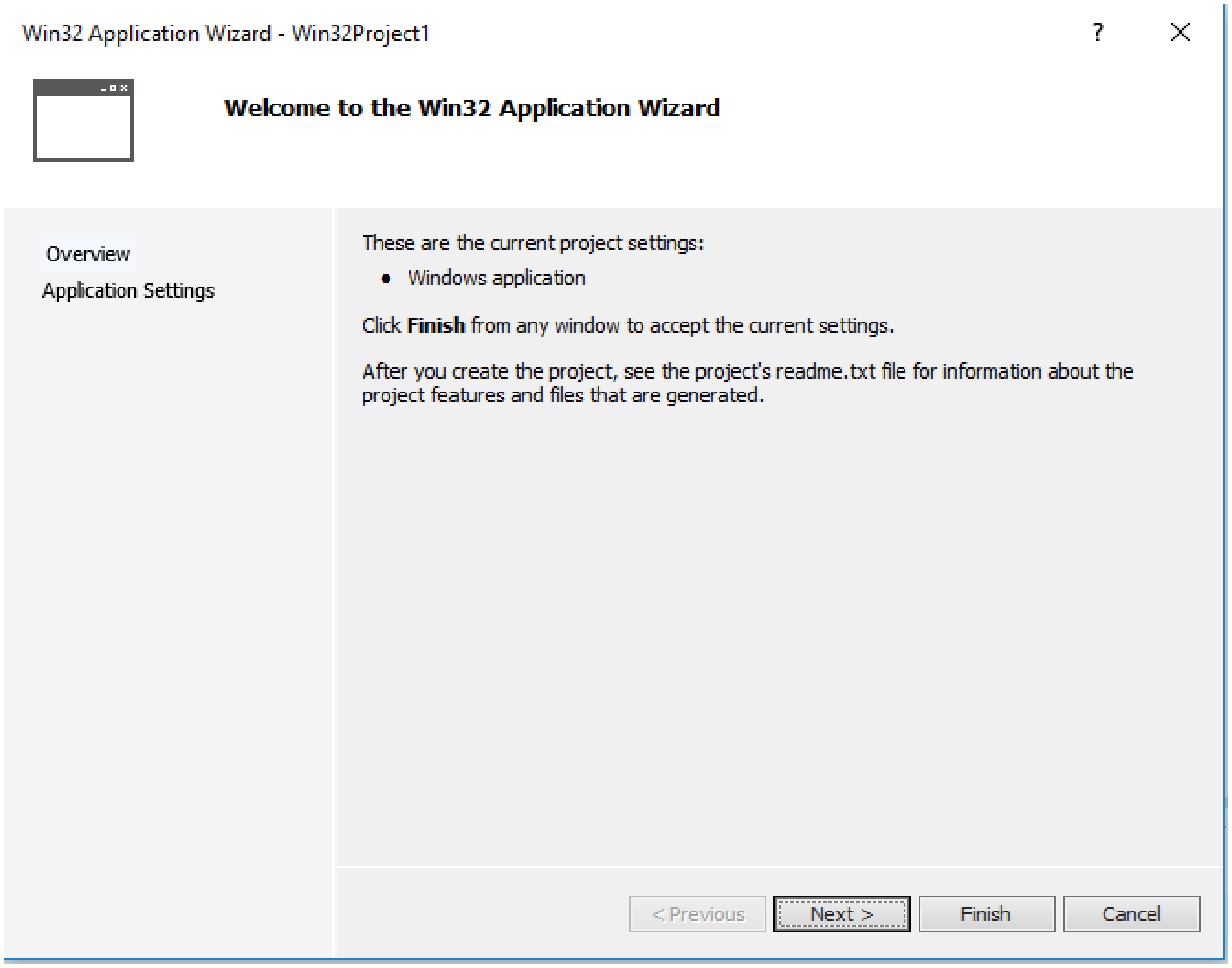
Solution: Create new solution

Solution name: Win32Project1

[✓] Create directory for solution

[ ] Add to source control

vii. After this, the Win32 Application wizard appears. Click "Next"


Win32 Application Wizard - Win32Project1

Welcome to the Win32 Application Wizard

These are the current project settings:

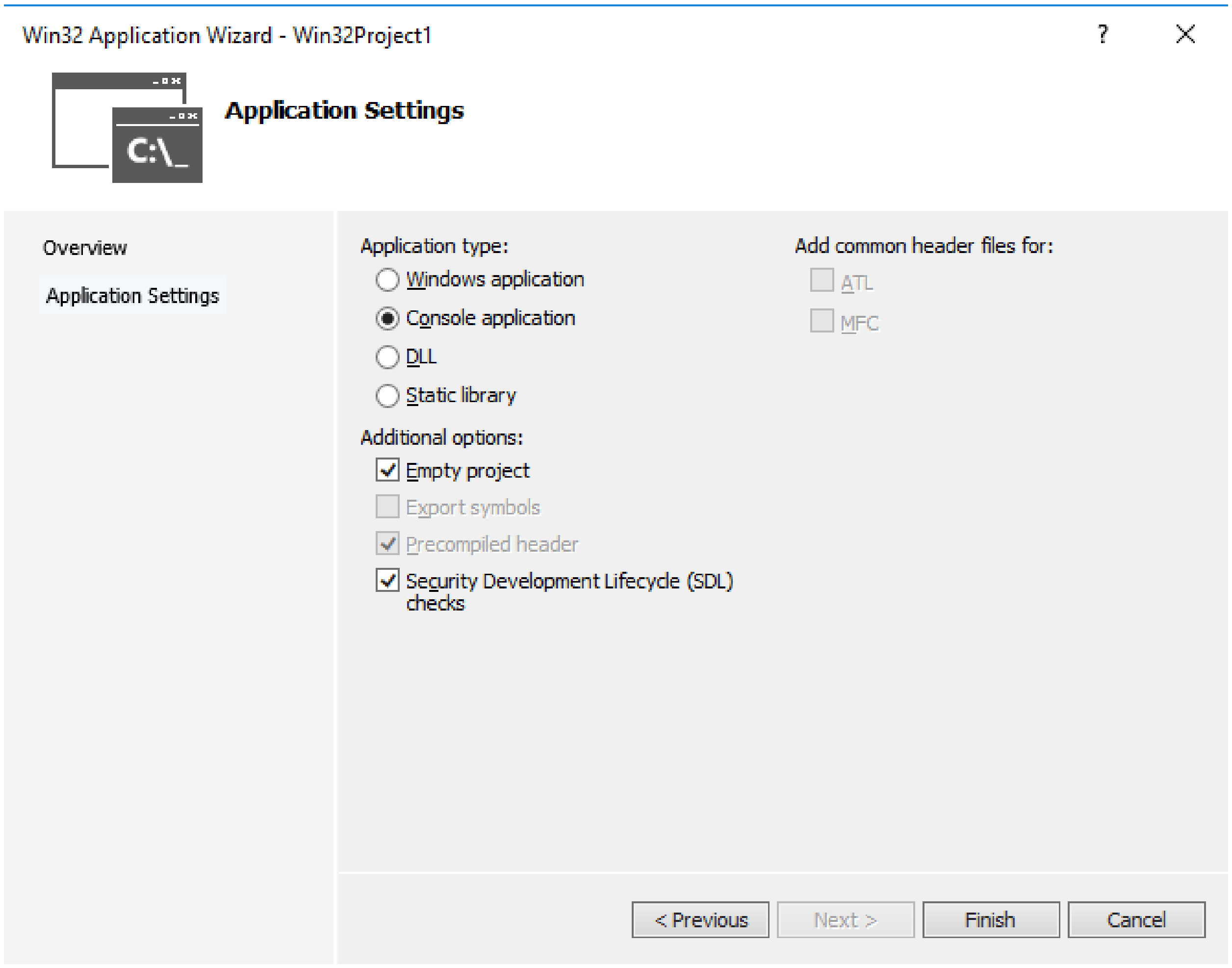
- Windows application

Click **Finish** from any window to accept the current settings.

After you create the project, see the project's readme.txt file for information about the project features and files that are generated.

viii. Select “Console Application” and “Empty Project”

ix. Click Finish


Win32 Application Wizard - Win32Project1

Application Settings

Overview
Application Settings

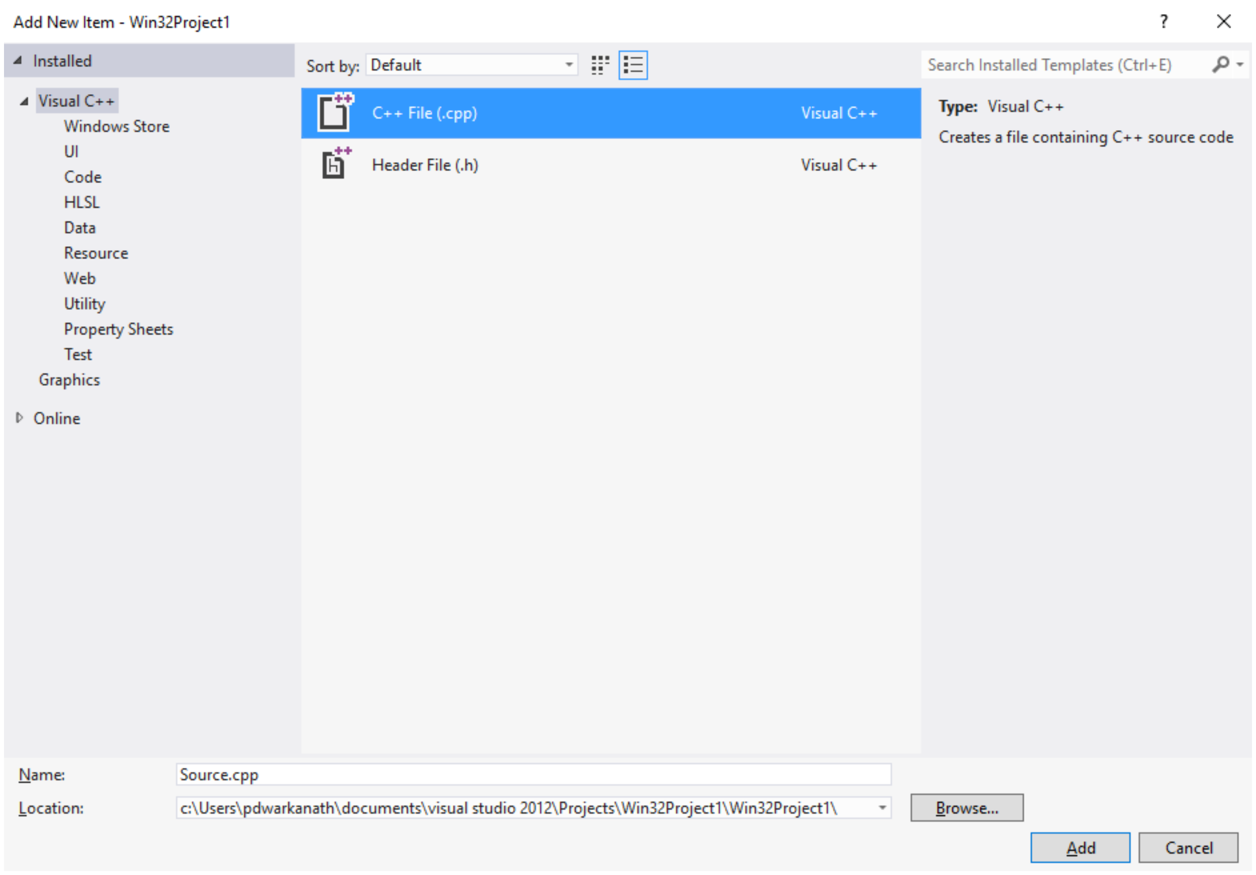
Application type:
- ( ) Windows application
- (●) Console application
- ( ) DLL
- ( ) Static library

Additional options:
- [✓] Empty project
- [ ] Export symbols
- [✓] Precompiled header
- [✓] Security Development Lifecycle (SDL) checks

Add common header files for:
- [ ] ATL
- [ ] MFC

b. Open a new C++ file

i. Select Project > Add New Item
ii. In the left pane, select Installed > Visual C++
iii. Select C++ file (.cpp)
iv. Change name if necessary
v. Click Add

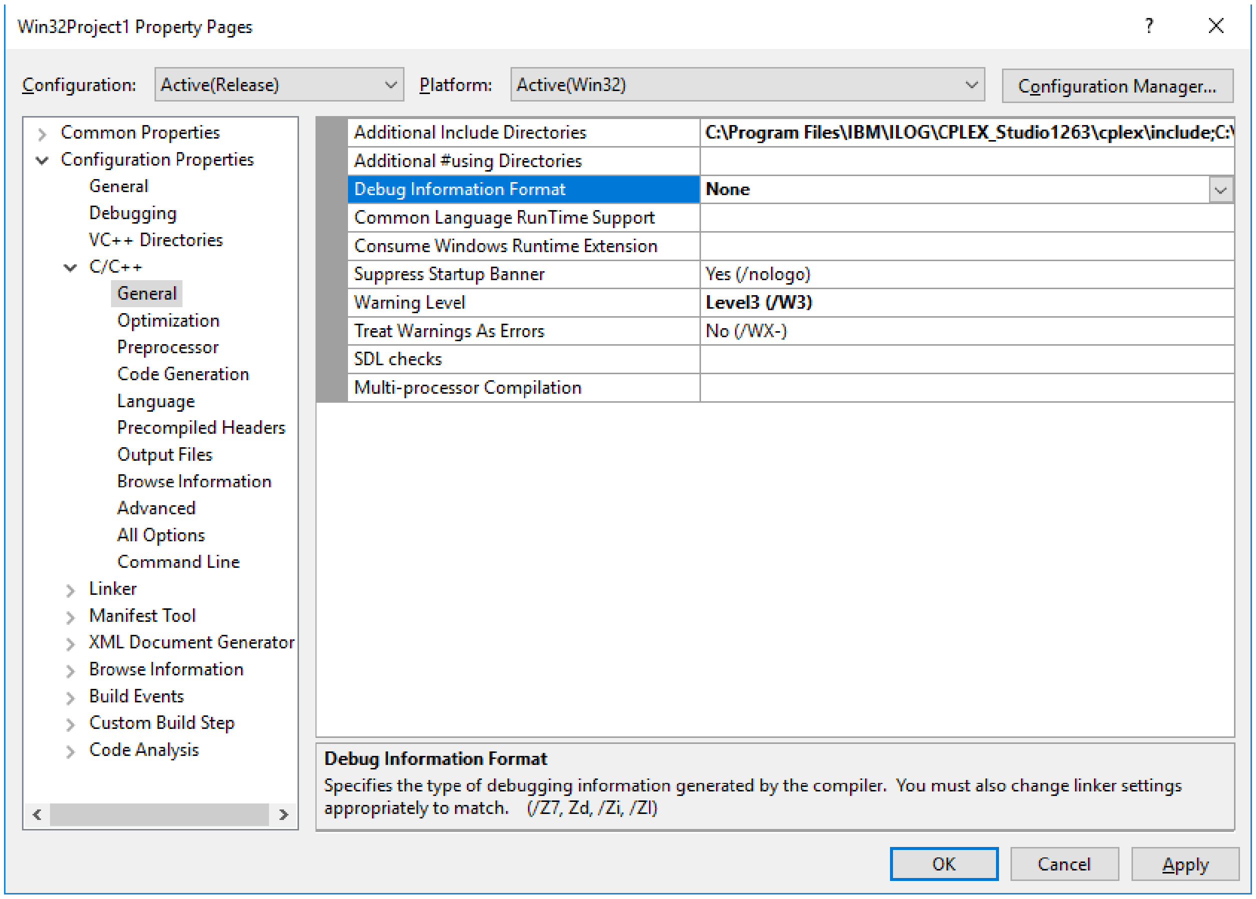

Add New Item - Win32Project1
Installed
Visual C++
Windows Store
UI
Code
HLSL
Data
Resource
Web
Utility
Property Sheets
Test
Graphics
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
C++ File (.cpp) Visual C++
Header File (.h) Visual C++
Type: Visual C++
Creates a file containing C++ source code
Browse...
Add
Cancel

Name: Source.cpp

Location: c:\Users\pdwarkanath\documents\visual studio 2012\Projects\Win32Project1\Win32Project1\

c. Link CPLEX and Concert to your project

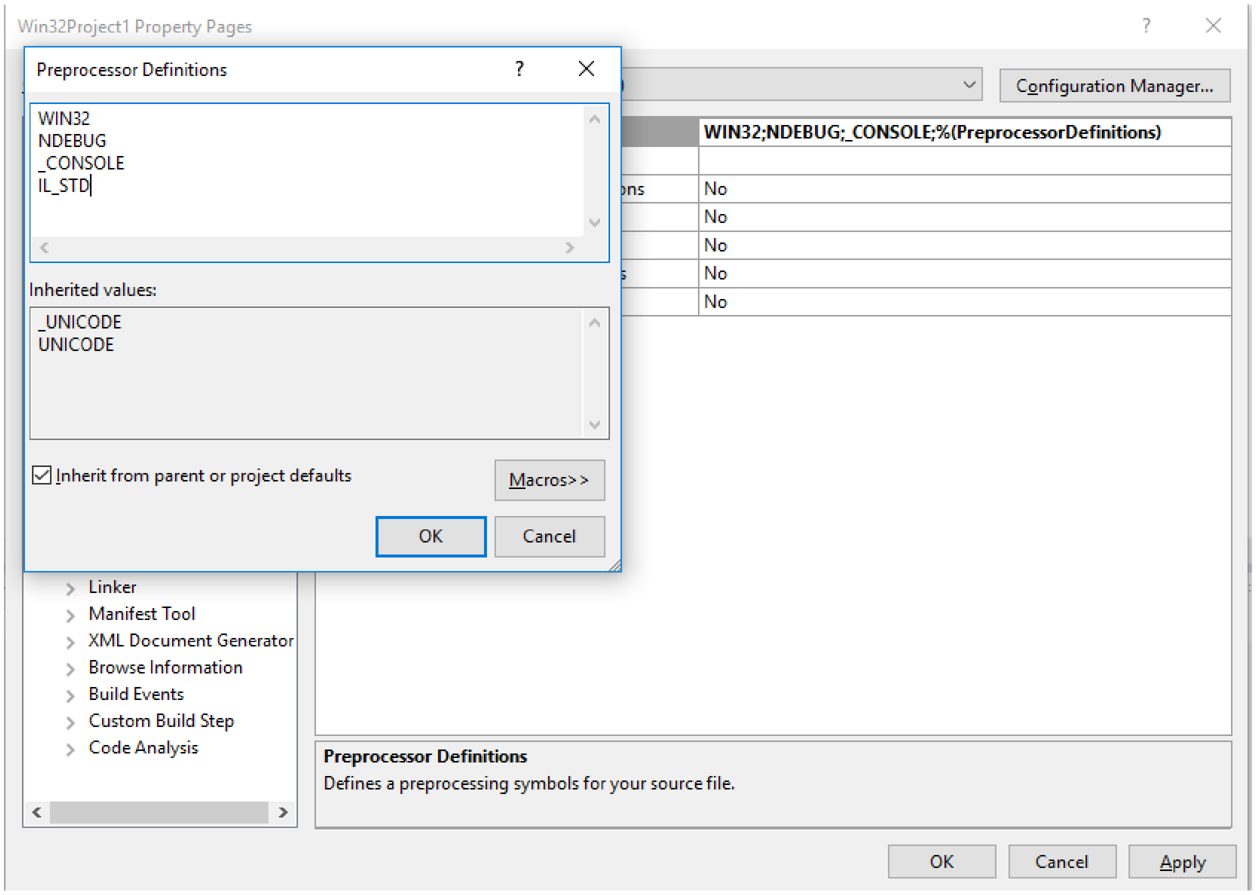
i. Select Project > Properties
ii. In the project properties box, select Configuration Properties > C++ > General
iii. Add these two pathnames:
   - <CPLEXDIR>\cplex\include
   - <CPLEXDIR>\concert\include
iv. Select SDL checks as “No”
v. Select Debug Information Format as “None”



Additional Include Directories	C:\Program Files\IBM\ILOG\CPLEX_Studio1263\cplex\include;C:\
Additional #using Directories	
Debug Information Format	None
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	Level3 (/W3)
Treat Warnings As Errors	No (/WX-)
SDL checks	
Multi-processor Compilation	

Debug Information Format
Specifies the type of debugging information generated by the compiler. You must also change linker settings appropriately to match. (/Z7, Zd, /Zi, /ZI)

vi. On the left pane, select Configuration Properties > C++ > Preprocessor
vii. Add IL_STD to the Preprocessor Definitions


Win32Project1 Property Pages

Preprocessor Definitions

```
WIN32
NDEBUG
_CONSOLE
IL_STD
```

Inherited values:

```
_UNICODE
UNICODE
```

☑ Inherit from parent or project defaults

WIN32;NDEBUG;_CONSOLE;%(PreprocessorDefinitions)

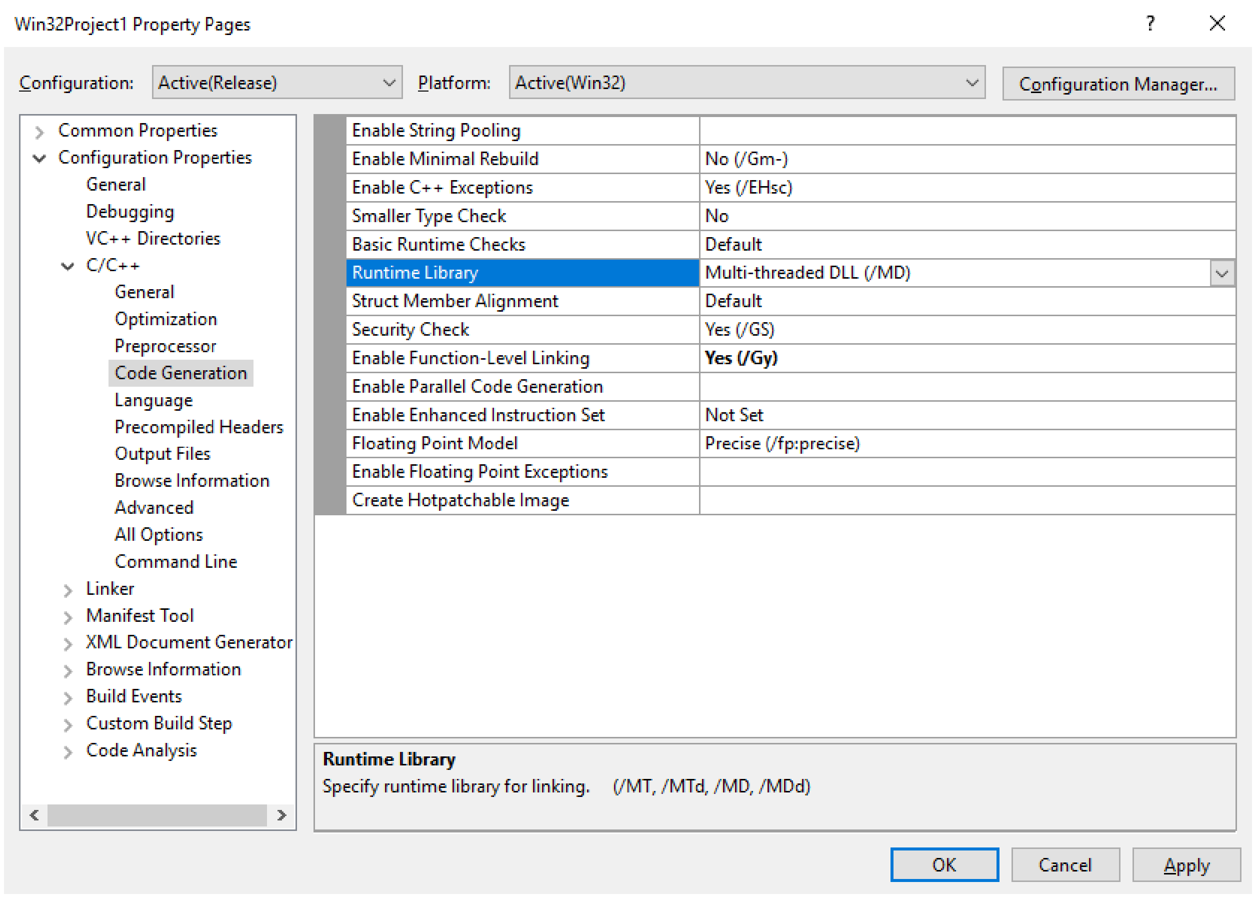
ons	No
	No
	No
s	No
	No

Preprocessor Definitions

Defines a preprocessing symbols for your source file.

viii. Select Configuration Properties > C++ > Code Generation

ix. Set Runtime Library to Multi-threaded DLL (/MD) (for pre-2012 versions of Visual Studio, use Multi-threaded (/MT))


Win32Project1 Property Pages

Configuration: Active(Release) Platform: Active(Win32) Configuration Manager...

- Common Properties
- Configuration Properties
  - General
  - Debugging
  - VC++ Directories
  - C/C++
    - General
    - Optimization
    - Preprocessor
    - Code Generation
    - Language
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Advanced
    - All Options
    - Command Line
  - Linker
  - Manifest Tool
  - XML Document Generator
  - Browse Information
  - Build Events
  - Custom Build Step
  - Code Analysis

Enable String Pooling	
Enable Minimal Rebuild	No (/Gm-)
Enable C++ Exceptions	Yes (/EHsc)
Smaller Type Check	No
Basic Runtime Checks	Default
Runtime Library	Multi-threaded DLL (/MD)
Struct Member Alignment	Default
Security Check	Yes (/GS)
Enable Function-Level Linking	**Yes (/Gy)**
Enable Parallel Code Generation	
Enable Enhanced Instruction Set	Not Set
Floating Point Model	Precise (/fp:precise)
Enable Floating Point Exceptions	
Create Hotpatchable Image	

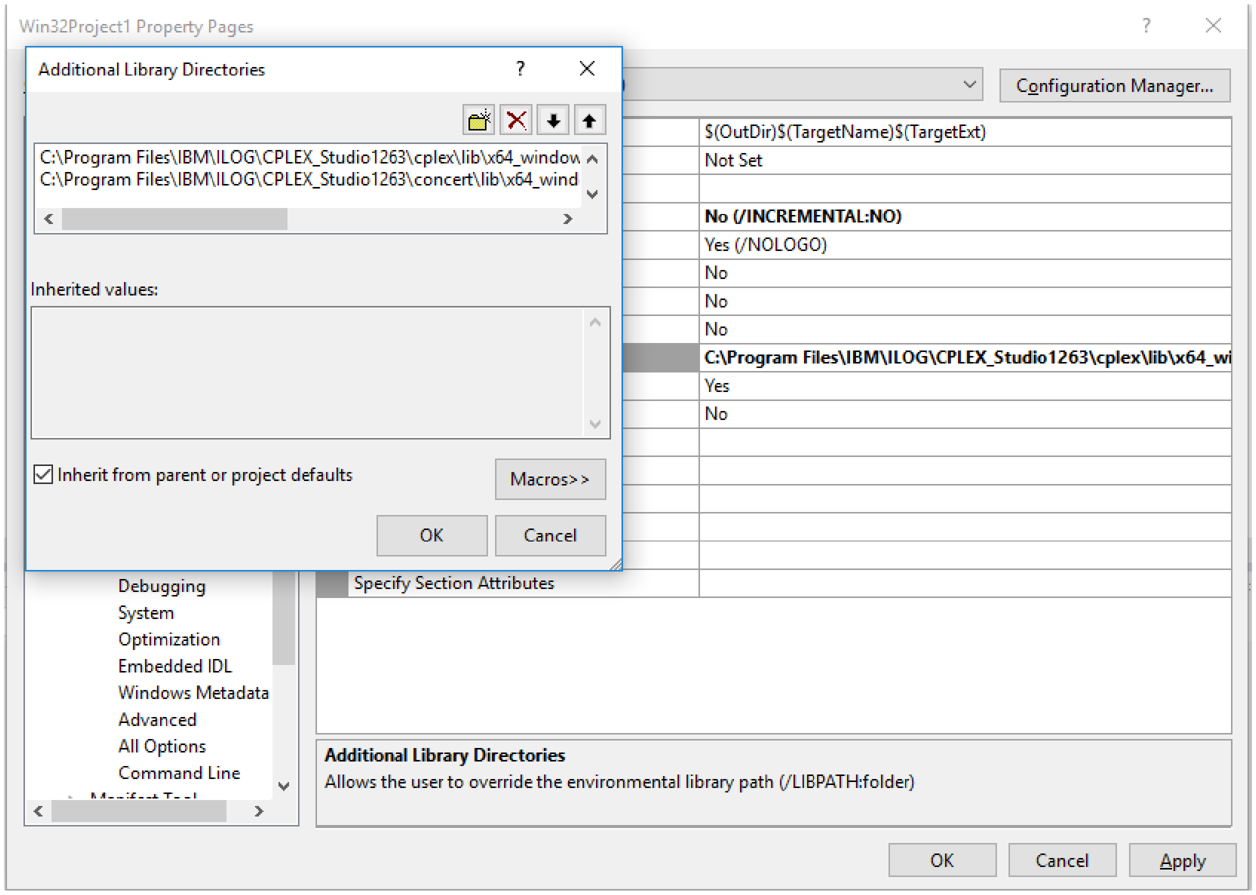
Runtime Library
Specify runtime library for linking. (/MT, /MTd, /MD, /MDd)

x. Select Configuration Properties > Linker > General

xi. In the “Additional Library Directories” add:

- <CPLEXDIR> \cplex\lib\x64_windows_vs2012\stat_mda
- <CPLEXDIR> \concert\lib\x64_windows_vs2012\stat_mda

(vs2012 can be replaced by vs20xx where 20xx is the Visual Studio edition. This tutorial was made with Visual Studio 2012 and hence uses vs2012)


Win32Project1 Property Pages

Additional Library Directories

```
C:\Program Files\IBM\ILOG\CPLEX_Studio1263\cplex\lib\x64_window
C:\Program Files\IBM\ILOG\CPLEX_Studio1263\concert\lib\x64_wind
```

Inherited values:

☑ Inherit from parent or project defaults

	$(OutDir)$(TargetName)$(TargetExt)
	Not Set
	No (/INCREMENTAL:NO)
	Yes (/NOLOGO)
	No
	No
	No
	C:\Program Files\IBM\ILOG\CPLEX_Studio1263\cplex\lib\x64_wi
	Yes
	No
Specify Section Attributes	

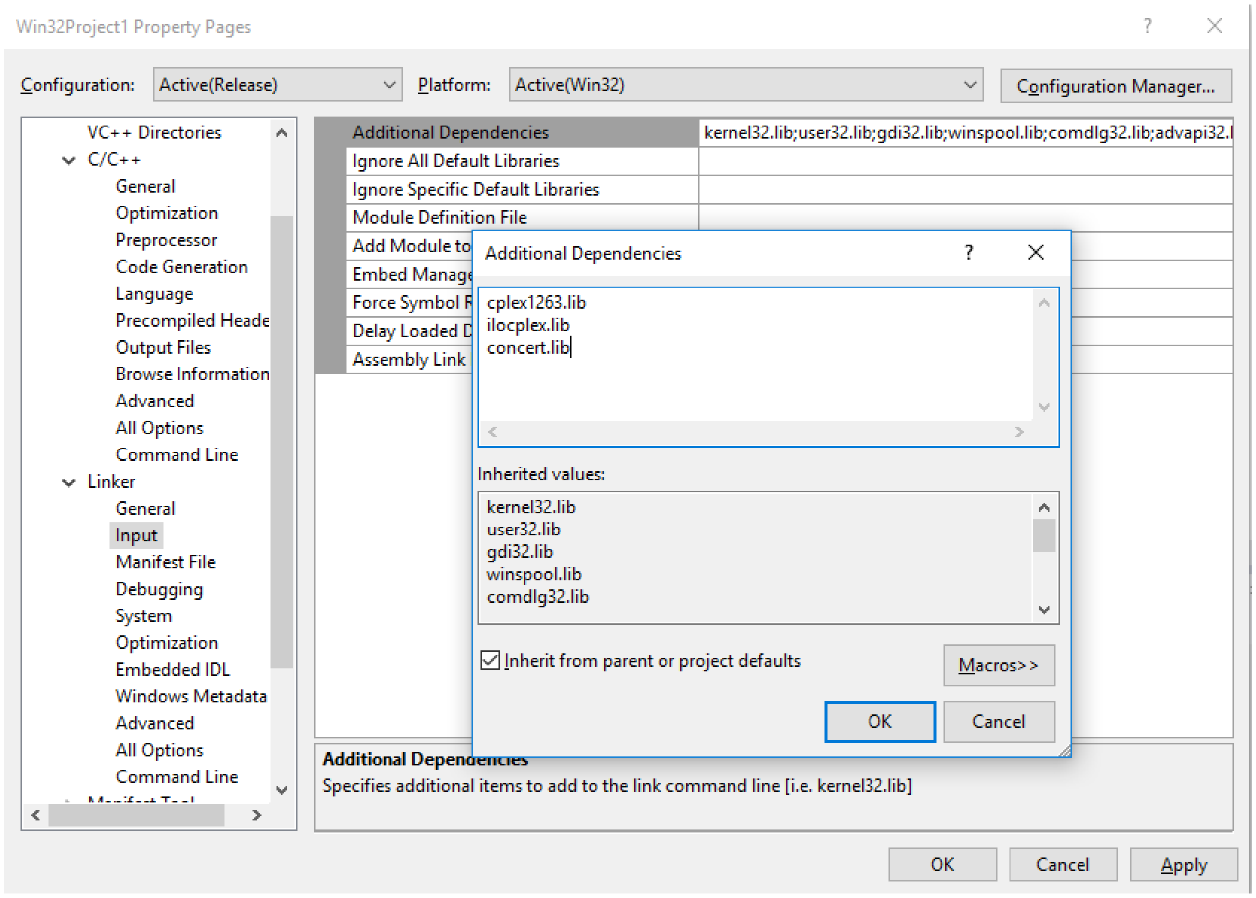
Additional Library Directories
Allows the user to override the environmental library path (/LIBPATH:folder)

xii. Select Configurations Properties > Linker > Input

xiii. In "Additional Dependencies" add:

- cplex1263.lib
- ilocplex.lib
- concert.lib

(cplex1263.lib can be replaced by cplexXXxx.lib where XXxx is the CPLEX version. This tutorial was made with CPLEX 12.63 and hence uses cplex1263)


Win32Project1 Property Pages

Configuration: Active(Release) Platform: Active(Win32) Configuration Manager...

Additional Dependencies	kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32.lib;advapi32.l
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	

Additional Dependencies

```
cplex1263.lib
ilocplex.lib
concert.lib
```

Inherited values:

```
kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib
```

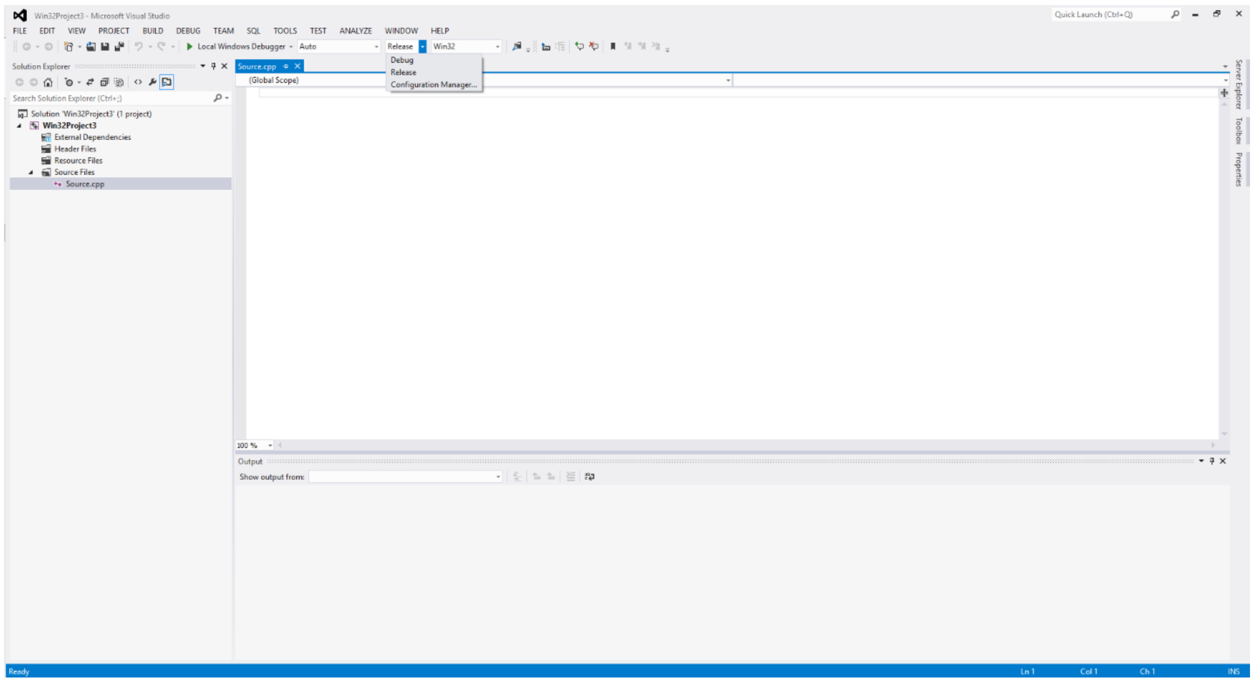
☑ Inherit from parent or project defaults

Additional Dependencies
Specifies additional items to add to the link command line [i.e. kernel32.lib]

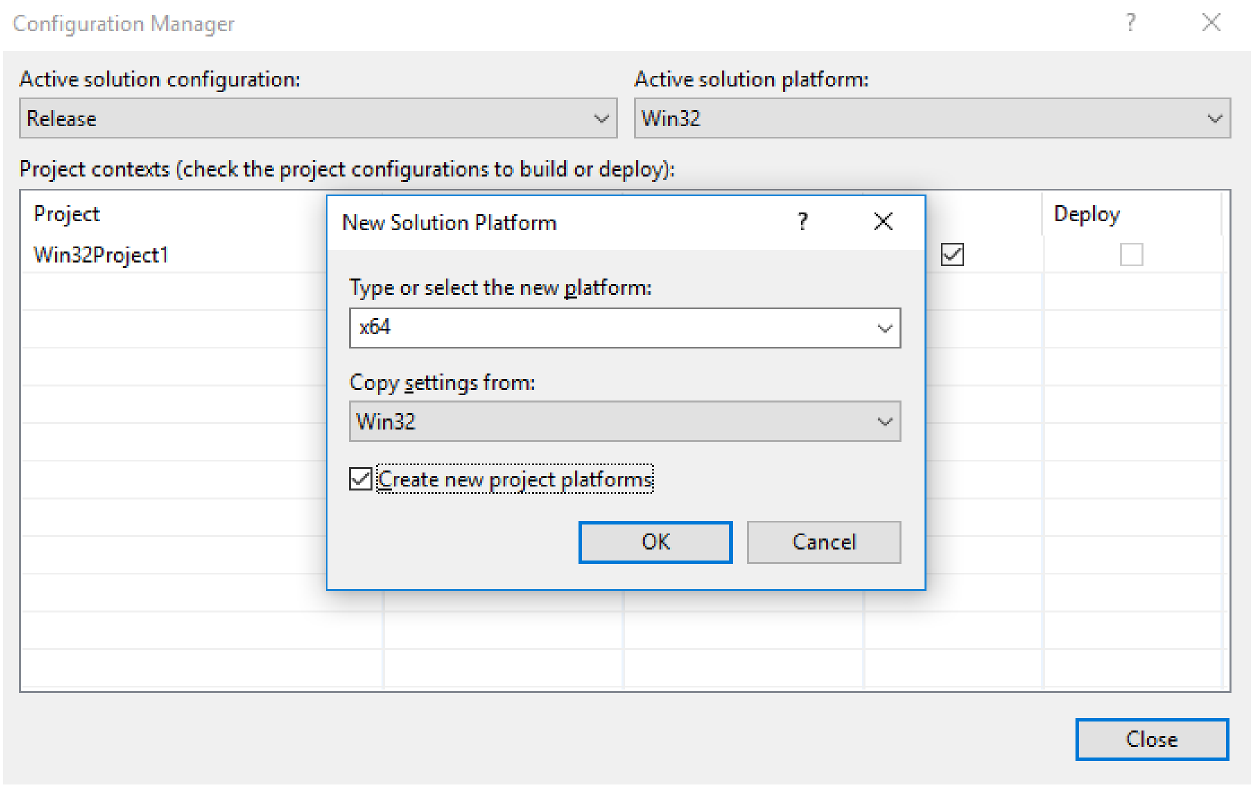
xiv. Click OK to close the Properties box

d. Change to win64 platform

i. From the top ribbon, select “Release” and then “Configuration Manager”



ii. Under Active solution platform, select <New>
iii. Select x64 under "Type or select the new platform:"
iv. Select Win32 under "Copy settings from:"
v. Keep the "Create new project platforms" option checked



Configuration Manager

Active solution configuration: Release

Active solution platform: Win32

Project contexts (check the project configurations to build or deploy):

Project	Deploy
Win32Project1	☐

New Solution Platform

Type or select the new platform: x64

Copy settings from: Win32

☑ Create new project platforms

vi. Click OK to close the box
vii. Click Close to close the Configuration Manager

e. Add CPLEX DLL file to project folder

i. Go to <CPLEXDIR>\cplex\bin\x64_win64 and copy the file cplex1263.dll
ii. Paste the file in the project folder <MYPROJDIR>\<ProjName>\<ProjName>. In the case of this tutorial the folder is:
C:\Users\pdwarkanath\Documents\Visual Studio 2012\Projects\Win32Project1\Win32Project1

4. Running a simple example program

Now that CPLEX is set up, we can run an example program to test it. A simple integer program is presented below:

Max: $x_1 + 0.64x_2$
s.t.: $50x_1 + 31x_2 \le 250$
$3x_1 - 2x_2 \ge -4$
$x_1, x_2 \in \mathbb{Z}_+$

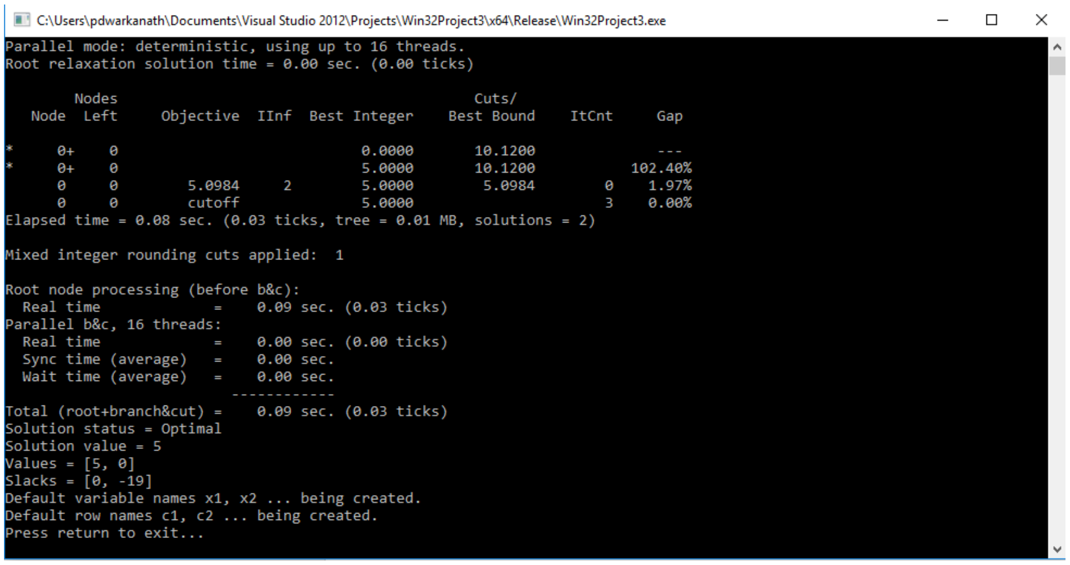
In order to solve this, it can be written in C++ as follows. The comments in green explain the steps below it.

```
#include <ilcplex/ilocplex.h>

ILOSTLBEGIN

static void
       populatebyrow (IloModel model, IloNumVarArray var, IloRangeArray con);

int main (void){

       // Setting up the CPLEX environment

       IloEnv env;

       // CPLEX solution block

       try {

              // Declaring the objects

              IloModel model(env);
              IloNumVarArray var (env);
              IloRangeArray con(env);
              populatebyrow(model, var, con);

              // Solving using CPLEX
```

```
		IloCplex cplex(model);
		cplex.solve();

		// Output solutions

		env.out() << "Solution status = " << cplex.getStatus() << endl;
		env.out() << "Solution value = " << cplex.getObjValue() << endl;

		IloNumArray vals(env);
		cplex.getValues(vals, var);
		env.out() << "Values = " << vals << endl;

		IloNumArray slacks(env);
		cplex.getSlacks(slacks, con);
		env.out() << "Slacks = " << slacks << endl;

		// Write model to file

		cplex.exportModel("ipex1.lp");

	}

	// Error handling

	catch (IloException e){
		cerr << "Concert exception caught: " << e << endl;
	}

	catch (...){
		cerr << "Unknown exception caught." << endl;
	}

	// End environment

	env.out() << "Press return to exit..." << endl;
	std::getchar();
	env.end();
	return 0;

}

static void
	populatebyrow (IloModel model, IloNumVarArray x, IloRangeArray c){

		// retrieve model environment

		IloEnv env = model.getEnv();

		// Define variables

		x.add(IloNumVar(env, 0.0, IloInfinity, ILOINT));
		x.add(IloNumVar(env, 0.0, IloInfinity, ILOINT));

		// Add objective

		model.add(IloMaximize(env, 1.00 * x[0] + 0.64 * x[1] ));

		// Add constraints
```

```
                c.add( 50 * x[0] + 31 * x[1] <= 250);
                c.add( 3 * x[0] - 2 * x[1] >= -4);
                model.add(c);
}
```

The optimal integer solution is (5, 0) with slacks (0, -19).



C:\Users\pdwarkanath\Documents\Visual Studio 2012\Projects\Win32Project3\x64\Release\Win32Project3.exe

```
Parallel mode: deterministic, using up to 16 threads.
Root relaxation solution time = 0.00 sec. (0.00 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                            0.0000       10.1200              ---
*     0+    0                            5.0000       10.1200           102.40%
      0     0        5.0984     2        5.0000        5.0984        0    1.97%
      0     0        cutoff              5.0000                      3    0.00%
Elapsed time = 0.08 sec. (0.03 ticks, tree = 0.01 MB, solutions = 2)

Mixed integer rounding cuts applied:  1

Root node processing (before b&c):
  Real time             =    0.09 sec. (0.03 ticks)
Parallel b&c, 16 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.09 sec. (0.03 ticks)
Solution status = Optimal
Solution value = 5
Values = [5, 0]
Slacks = [0, -19]
Default variable names x1, x2 ... being created.
Default row names c1, c2 ... being created.
Press return to exit...
```

5. Running another example program

The previous example is a very simple use case for CPLEX. Linear programming problems can run into 100s and even thousands of variables. In the first case, there were only 2. So, it was possible to write the coefficients for constraints and objective functions directly in the code. However, as the problem becomes larger, this becomes tedious.

C++ allows a programmer to separately read a file with all the coefficients and build large expressions for objective functions and constraints using for loops. The next example will deal with a slightly larger problem with 9 variables. The necessary data will be stored in a separate file and will be read using C++ code.
Problem:

Assume you run a power supply company. You have 9 power generators available, each of which has a minimum and maximum production level and a cost per unit output. The question is which generators to use in order to minimize the overall operation cost while satisfying the demand of 187 MW.

Generator no.	Min Output	Max Output	Cost
0	12	22	13
1	12	22	13
2	15	23	13
3	17.8	27.8	9.5

4	17.8	27.8	9.5
5	17.9	28.8	9.3
6	19	29	7.2
7	19	29	7.2
8	19	29	7.2

Model:

Decision variables:

- Let x_i be the output from generator i in MW for all $i \in \{0, \dots, 8\}$

Parameters:

- Let a_i be the minimum output from generator i in MW for all $i \in \{0, \dots, 8\}$
- Let b_i be the maximum output from generator i in MW for all $i \in \{0, \dots, 8\}$
- Let c_i be the cost of producing of power from generator i in \$/MW for all $i \in \{0, \dots, 8\}$
- Let d be the demand to be met

Objective Function:

$$\text{Min: } \sum_{i=0}^{8} c_i\, x_i$$

Constraints:

- Upper and lower bounds: $a_i \leq x_i \leq b_i$ for all $i \in \{0, \dots, 8\}$
- Demand constraint: $\sum_{i=0}^{8} x_i \geq d$

In order to solve this, it can be written in C++ as given below. The data required for this code to run is available in a C++ friendly format in the file “rates.dat” (Click here to download). The comments in green explain the steps below it.

IMPORTANT: Make sure that the “rates.dat” file is available in the <ProjectName>\<ProjectName> or the same folder in which the C++ file (extension: .cpp) is in the project directory.

```
#include <ilcplex/ilocplex.h>

ILOSTLBEGIN

int main(void){
	IloEnv env;
	try {
		// Declare parameters

		IloNumArray minArray(env), maxArray(env), cost(env);
		IloNum demand;

		// Read data file
```

```
        ifstream in("rates.dat");
        in >> minArray >> maxArray >> cost >> demand;

        // Create model

        IloModel mdl(env);

        // Define variables

        IloNumVarArray production(env);
        IloInt generators = minArray.getSize();

        // Set upper and lower bounds for variables

        for (IloInt j = 0; j < generators; j++) {
               production.add(IloNumVar(env, minArray[j], maxArray[j]));
        }

        // Build objective function expression

        mdl.add(IloMinimize(env, IloScalProd(cost, production)));

        // Demand constraint

        mdl.add(IloSum(production) >= demand);

        // Solve and output solutions to a file

        IloCplex cplex(mdl);
          cplex.exportModel("rates.lp");
          ofstream f_out("rates.sol");
          if (cplex.solve()) {

                f_out << "Solution status: " << cplex.getStatus() << endl;
                for (IloInt j = 0; j < generators; j++) {
                      f_out << "generator " << j << ": "
                                           << cplex.getValue(production[j]) << endl;
                }
                f_out << "Total cost = " << cplex.getObjValue() << endl;
          }
          else {
                 f_out << "No solution" << endl;
          cplex.printTime();
    }
}

// Error handling

catch (IloException& ex) {
       cerr << "Error: " << ex << endl;
}

catch (...) {
       cerr << "Error" << endl;
}

// End environment
```

```
        env.end();
        return 0;
}
```

The optimal solution is:

```
generator 0: 12
generator 1: 12
generator 2: 15
generator 3: 17.8
generator 4: 17.8
generator 5: 25.4
generator 6: 29
generator 7: 29
generator 8: 29

Total cost = 1707.82
```

This solution will be saved in a file named “rates.sol” in the <ProjectName>\<ProjectName> folder.

Appendix – Files and Code

The files used in this tutorial and the code snippets can be downloaded from the following links:

- Example 1 C++ Code: https://dwarkanath.com/example1.cpp
- Example 2 data file: https://dwarkanath.com/rates.dat
- Example 1 C++ Code: https://dwarkanath.com/rates.cpp
- Example 2 solution file: https://dwarkanath.com/rates.sol

References:

- Kianfar, K., Bansal, M. (2013). *CPLEX Concert Technology using C++*
- Rodriguez-Carbonell, E. (2017). *Tutorial on CPLEX Linear Programming*
- *Using IBM ILOG CPLEX optimizers with Microsoft Visual C++* (c_cpp.html)