Yuan Ze University

Gurobi Implementation

2015 summer

lobal Logistics Lab.

ver. 1.0.2

Outlines

- Introduction
- Installing Gurobi
- Creating a visual studio C++ project for Groubi
- Building and solving Gurobi model

Introduction (1/4)

- Gurobi is a state-of-the-art solver engine for optimization problems, including
  - Linear Problem (LP)
  - Mixed-Integer Linear Programming (MILP)
  - Quadratic Problem (QP) and Mixed-Integer Quadratic Problem (MIQP) (Gurobi 4.0 and later version)
  - Quadratically constrained programming (QCP) and Mixed-integer quadratically constrained programming (MIQCP) (Gurobi 5.0 and later version)
- Gurobi supports **parallel computing** for the modern **multi-core PCs**. It also offers **Groubi Cloud** on the Amazon Elastic Computing Cloud (EC2).

Introduction (2/4)

- Gurobi provides different interfaces for different users:
  - Gurobi Command Line
  - Gurobi Interactive Shell
  - programming language:
    - ➡C, C++, C#, Java, Python, VB, MATLAB or R.
  - Different modeling systems: **AMPL, GAMS, AIMMS, Microsoft Solver Foundation**, and etc.

Introduction (3/4)

- Groubi supports most platforms including Windows, Linux, and Mac OS X.
- The platforms for Gurobi Optimizer 6.0 include:

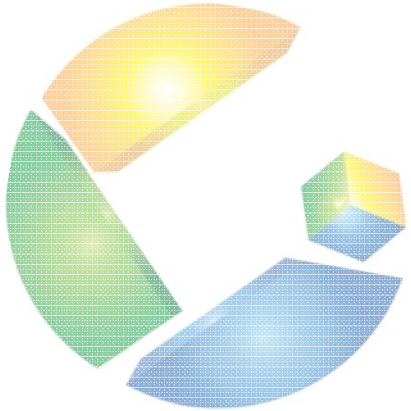
platform	Operating System	Compiler
Windows 32-bit (win32) Windows 64-bit (win64)	Windows Vista, Windows 7, Windows 8.1 and Windows Server 2008 R2	Visual Studio 2010, 2012, 2013
Linux 64-bit (linux64)	Red Hat, SUSE, Ubuntu	GCC 4.1, 4.3, 4.4, 4.6
Mac OS 64-bit (mac64)	Mac OS X 10.7 - 10.10	Xcode 4, 5, 6

Introduction (4/4)

- Groubi offers flexible software licensing strategies in different license types, including
  - Free evaluation license (all the features and power).
  - Commercial licenses: *named-user*, *single-use*, *unlimited-use*, *compute server*, *on demand cloud*, and *prepaid cloud* licenses.
  - **Free academic license.**

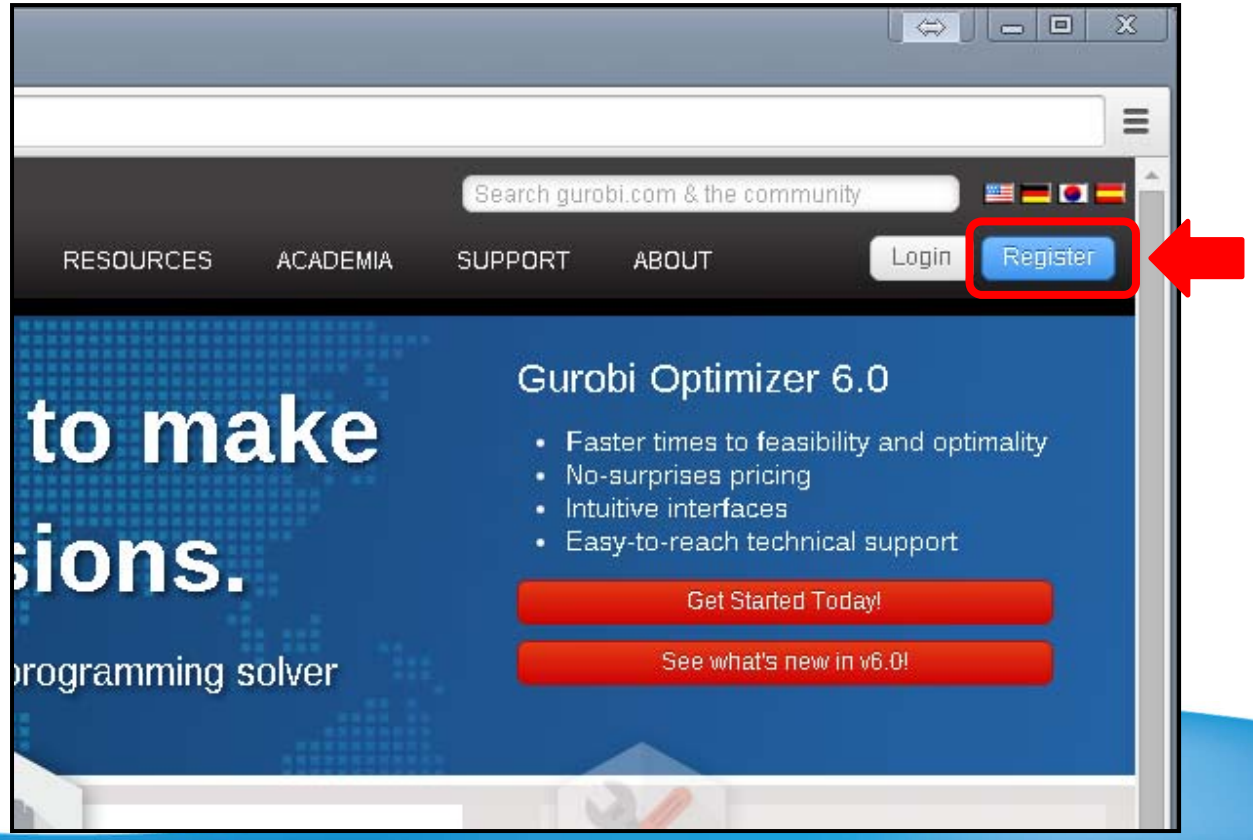
- For **free academic license**, you need to update your license every six months, and one license only for one PC.

Installation



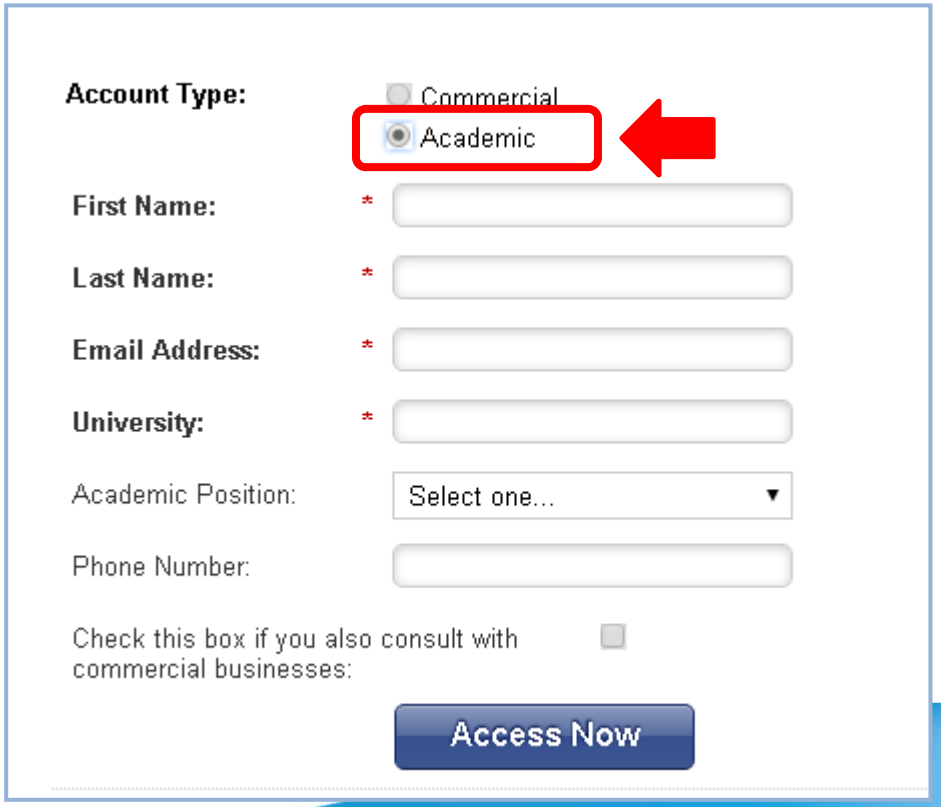
Installation (1/8)

- Go to Gurobi's website: http://www.gurobi.com/
- Register an Account



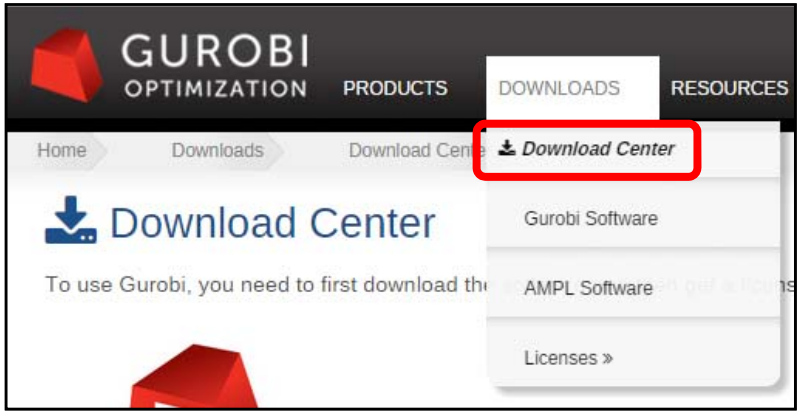
Installation (2/8)

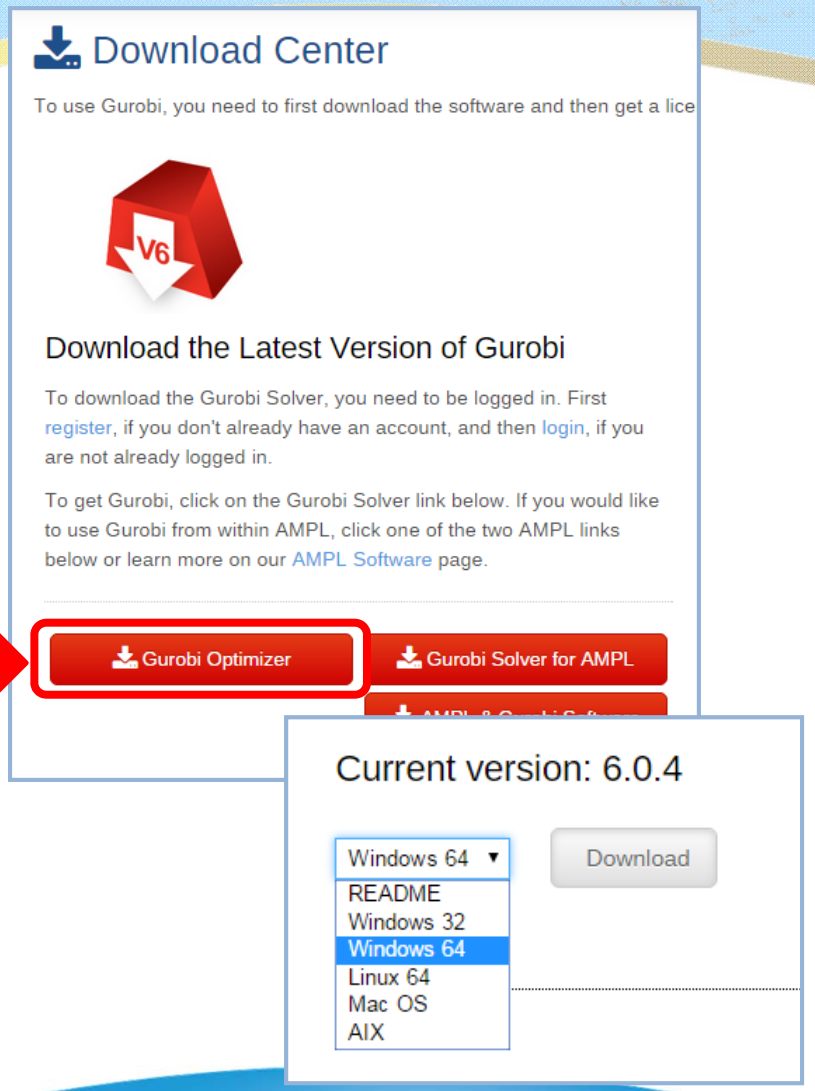
- Fill out the form and submit, then you can activate your account and get the password from e-mail.



Installation (3/8)

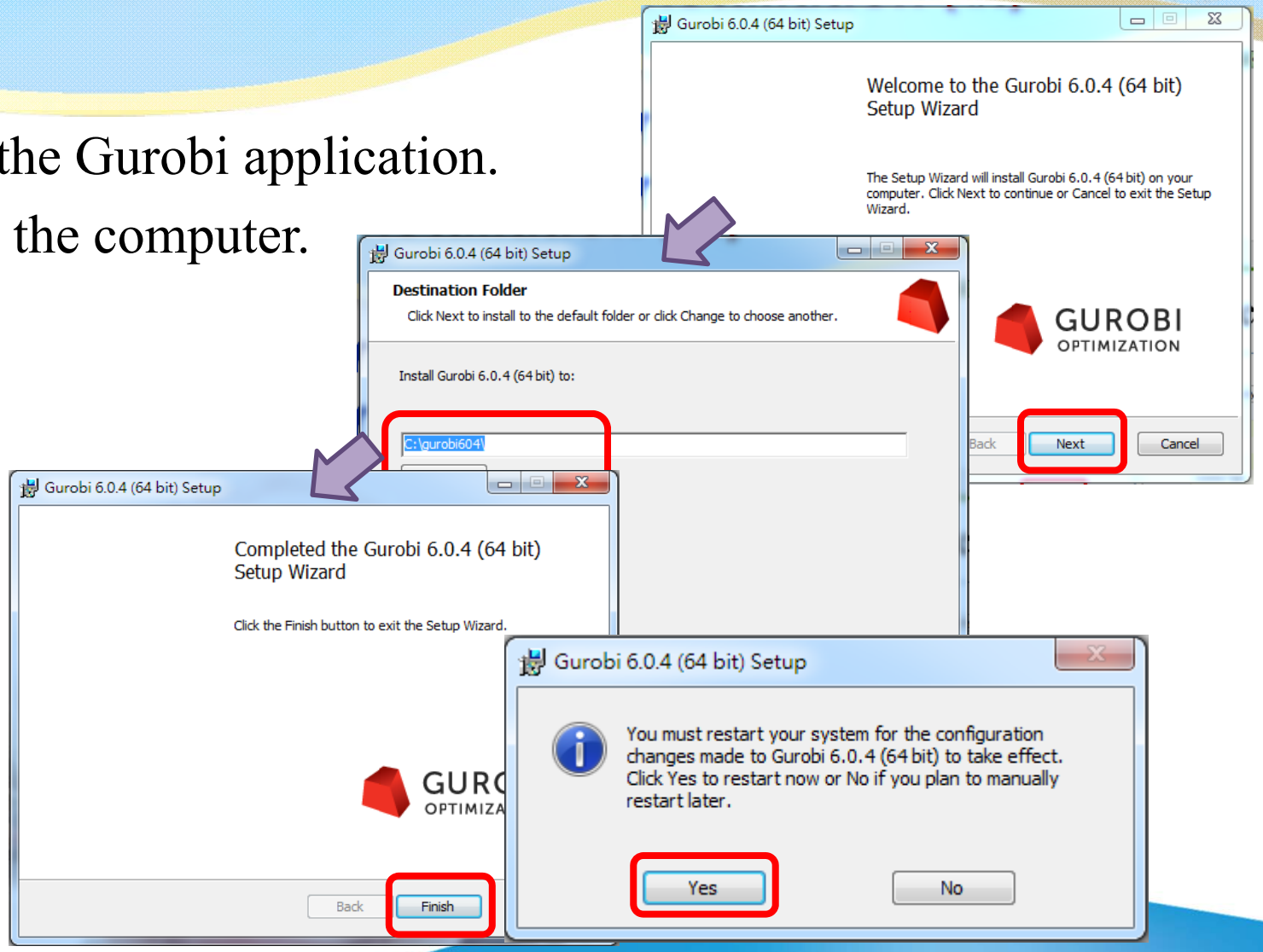
- Login your account, and go to Download Center to download the newest version of Gurobi Optimizer according to your PC's platform.



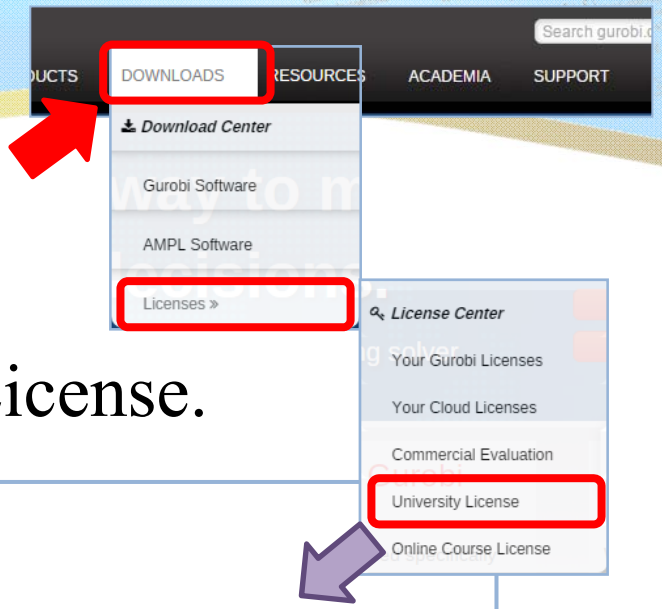


Installation (4/8)

- Install the Gurobi application.
- Restart the computer.



Installation (5/8)



- Login you Gurobi's account, and request for a Free Academic License.

Free Academic License

Request a free academic license

To request a free academic license, please read and accept the End User License Agreement.

End User License Agreement (View in PDF)
I accept the End User License Agreement

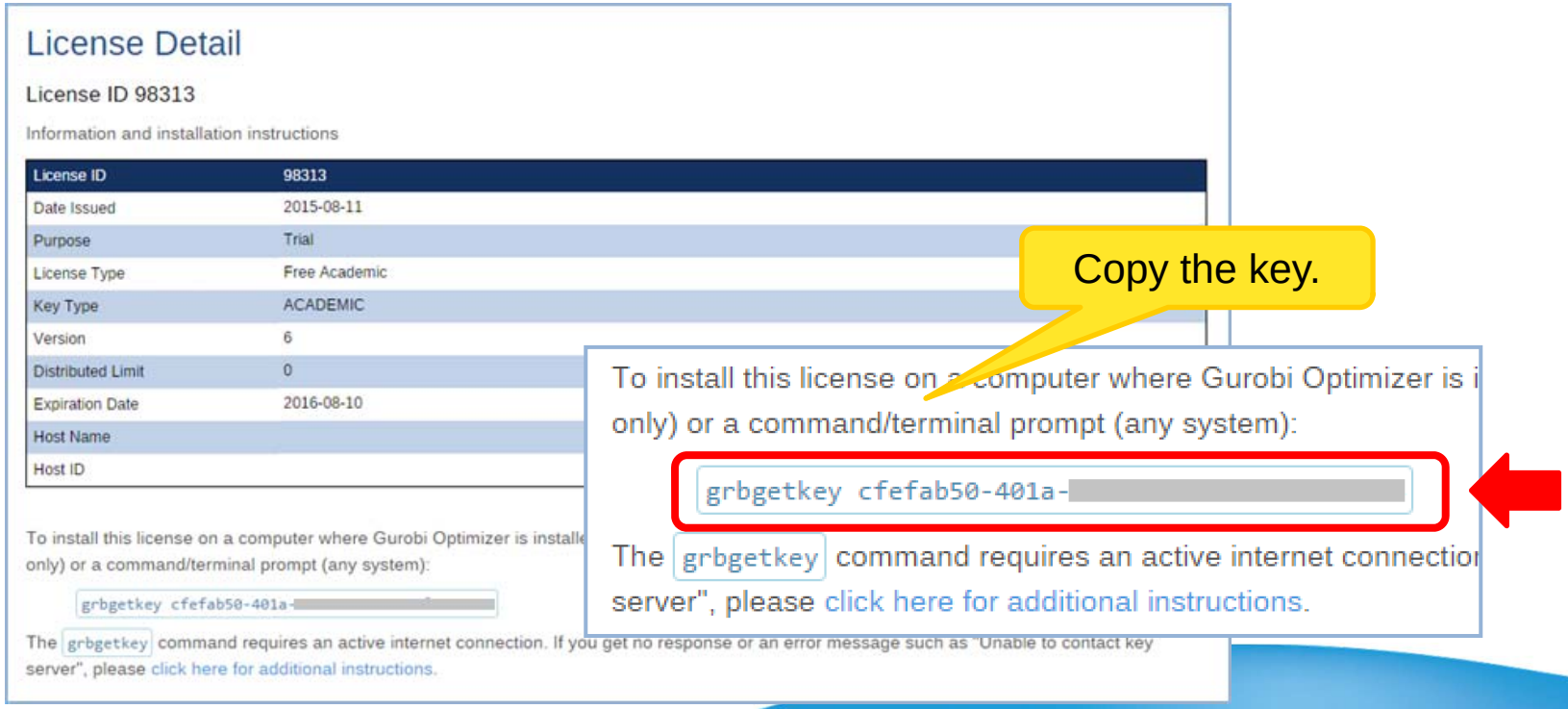
Conditions for the use of an Academic License: An academic license may only be used by a faculty member, a student, or a member of the research or administrative staffs of a degree-granting academic institution. The code may be used only for research and educational purposes. Access for commercial purposes is forbidden.
I accept these conditions

We urge academic users to upgrade to the latest version of Gurobi Optimizer. Some features, such as `grbgetkey`, may not work correctly in older releases.

Request License

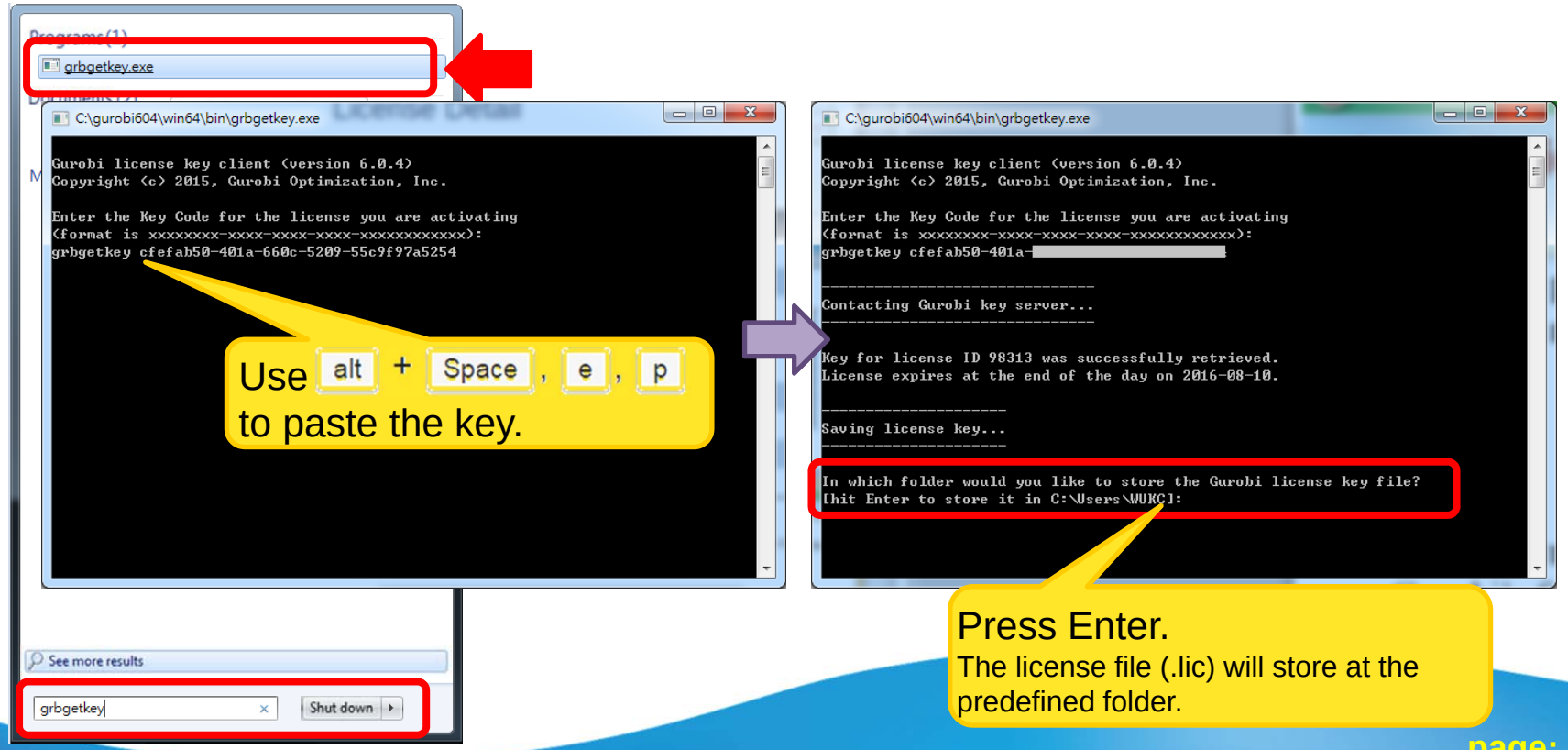
Installation (6/8)

- Login you Gurobi's account,
  and request for a Free Academic License.



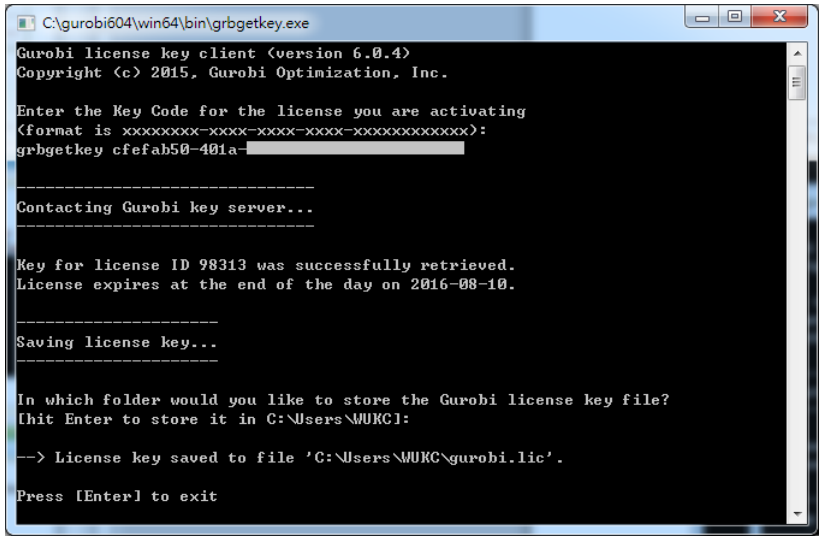
Installation (7/8)

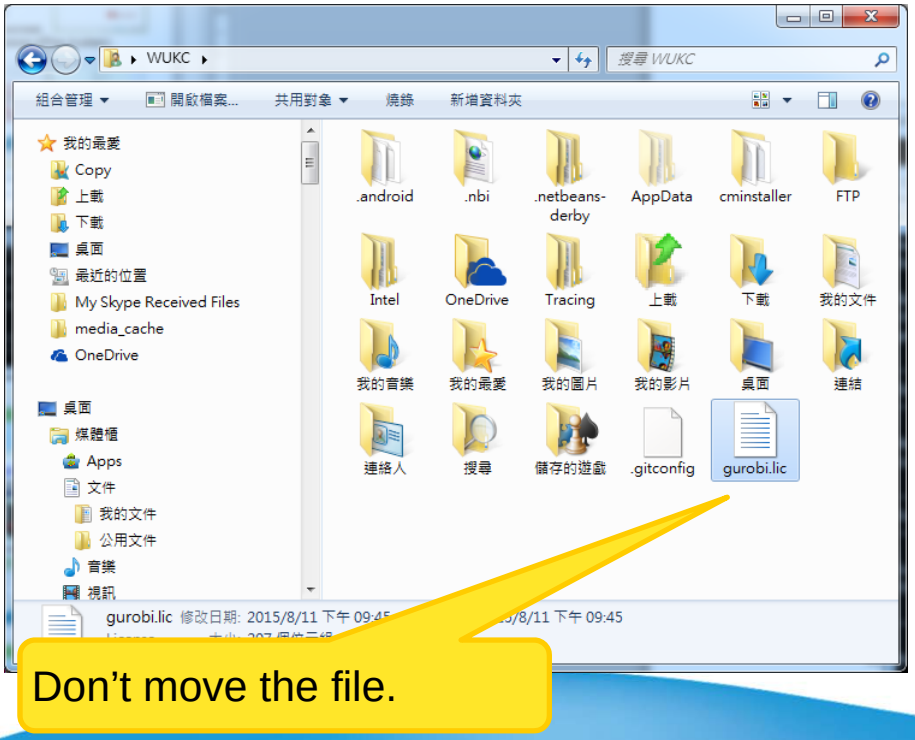
- Copy and paste grbgetkey and lincense and to the Search box on the Start menu (Windows only).



Installation (8/8)

- Copy and paste grbgetkey and lincense and to the Start/Run menu (Windows only).





Installation [additional] (1/4)

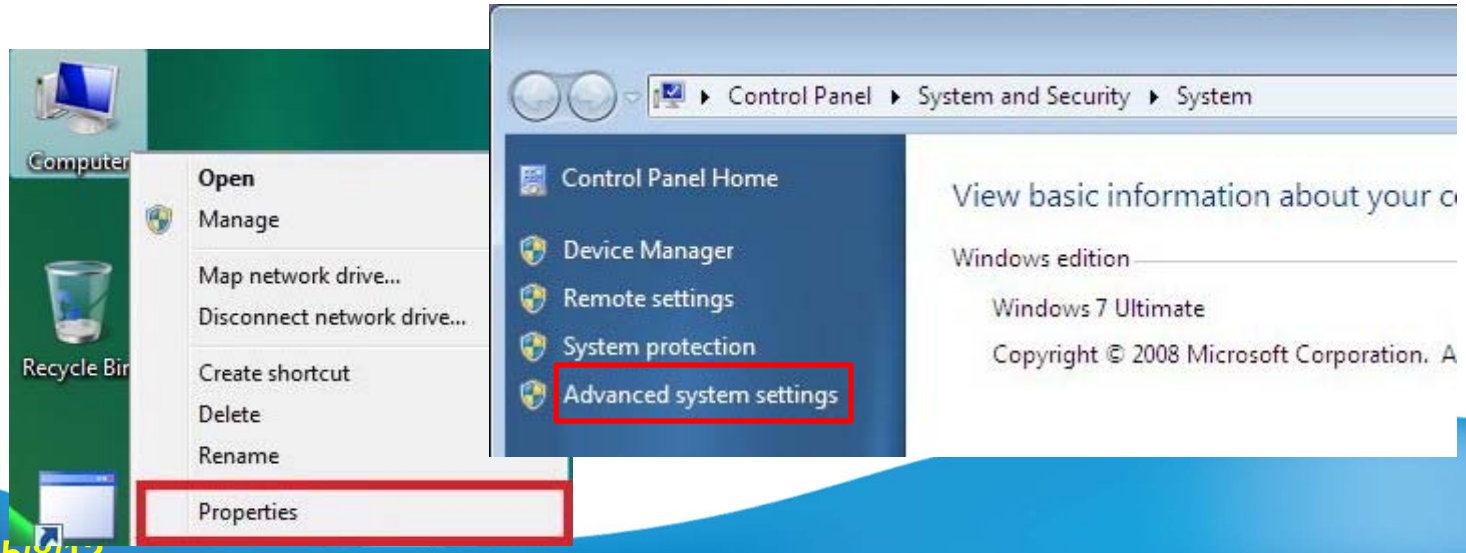
- When a C++ program uses the Gurobi solver, but an error with code number 10009 is shown. Then, (1) check the expired day of the license, or (2) check the location of Gurobi.

Installation [additional] (2/4)

- The default location for Gurobi is in C drive (C:\gurobixxx). If you install Gurobi to other drives, the environment variables of Windows needs to be modified.

1) Right click on the Computer icon and choose Properties option.

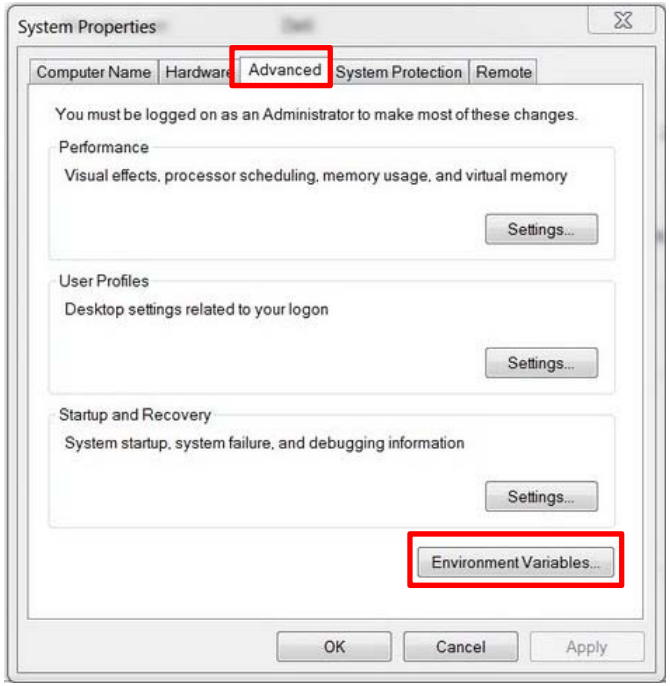
2) Click on Advanced system settings in the left pane

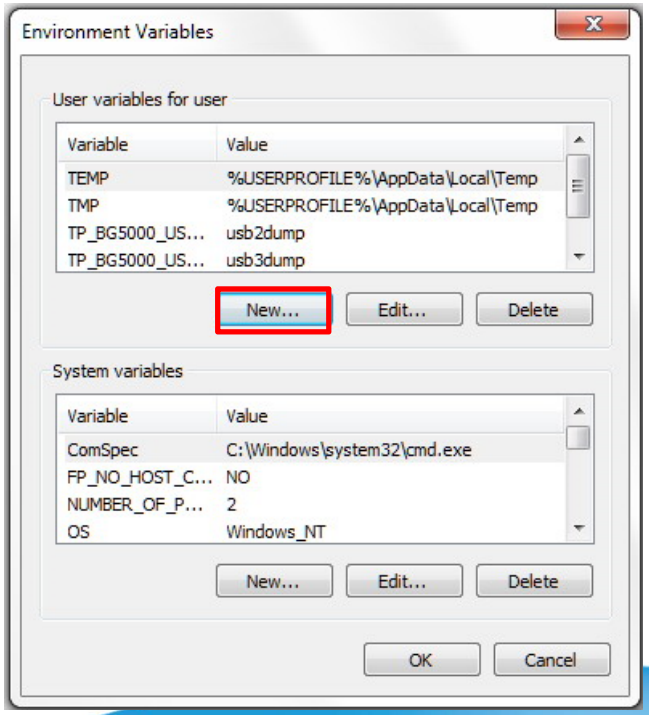


Installation [additional] (3/4)

3) Select Advanced tab and click on Environment Variables

4) Add a new User variable click on New button.



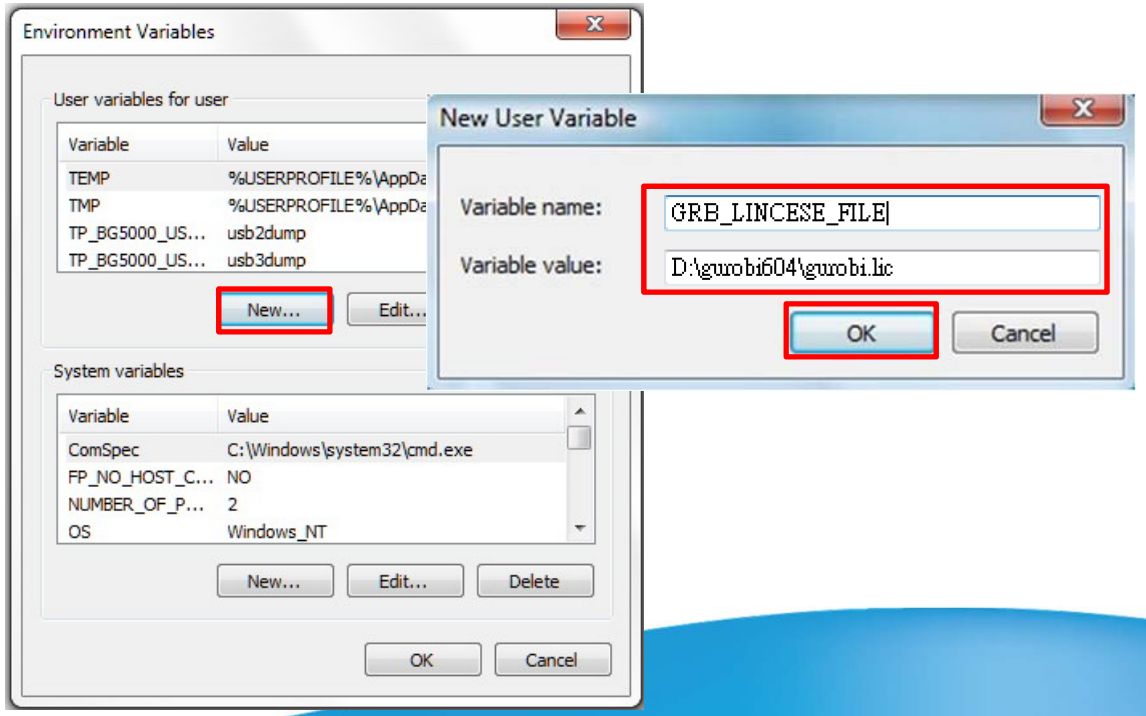


Installation [additional] (4/4)

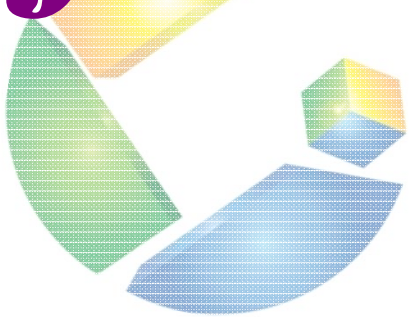
5) Give Variable name = GRB_LICENSE_FILE

and Variable value = D:\gurobi604\gurobi.lic (location of the file)

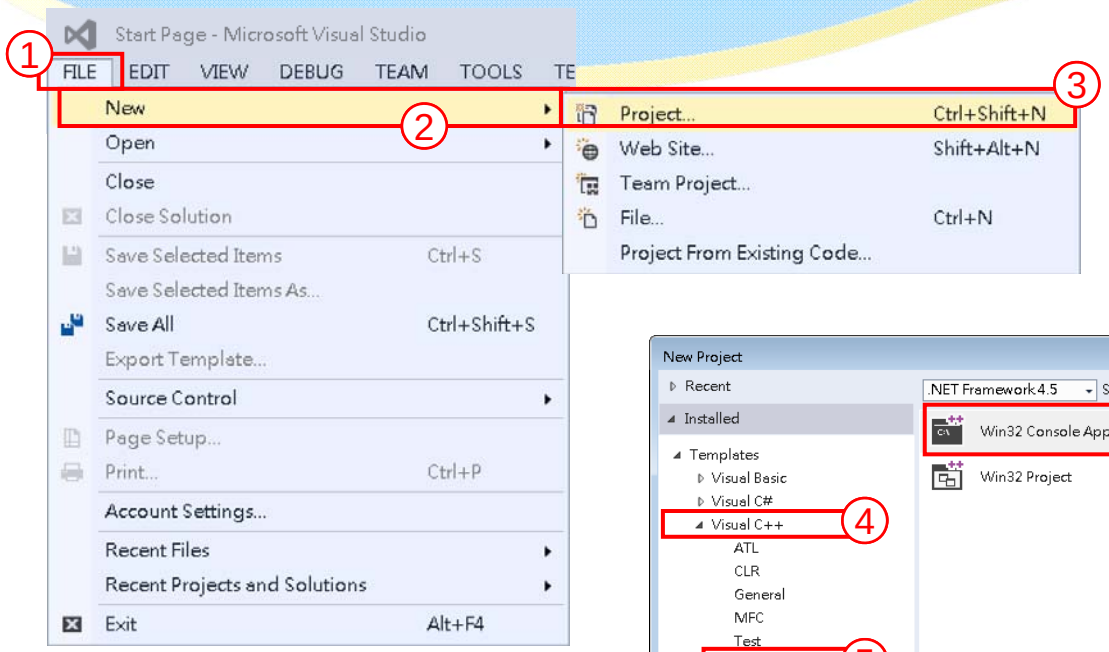
6) Restart your Visual Studio!



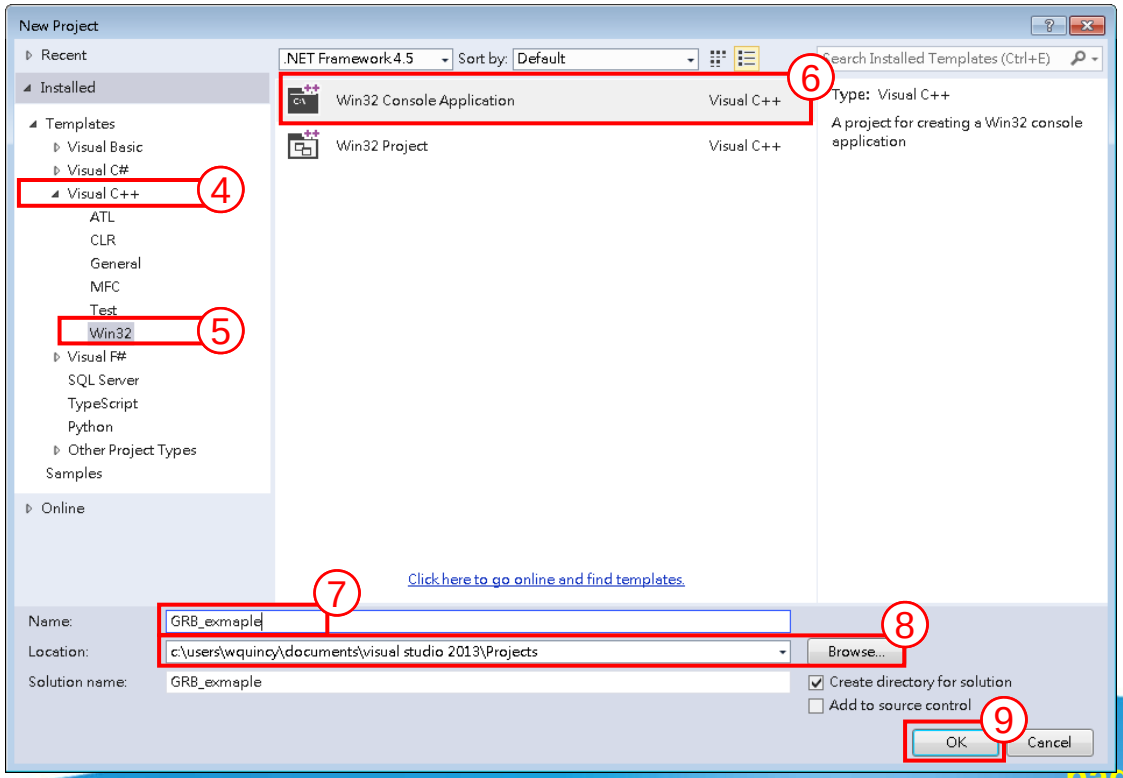
Creating Visual C++ Project



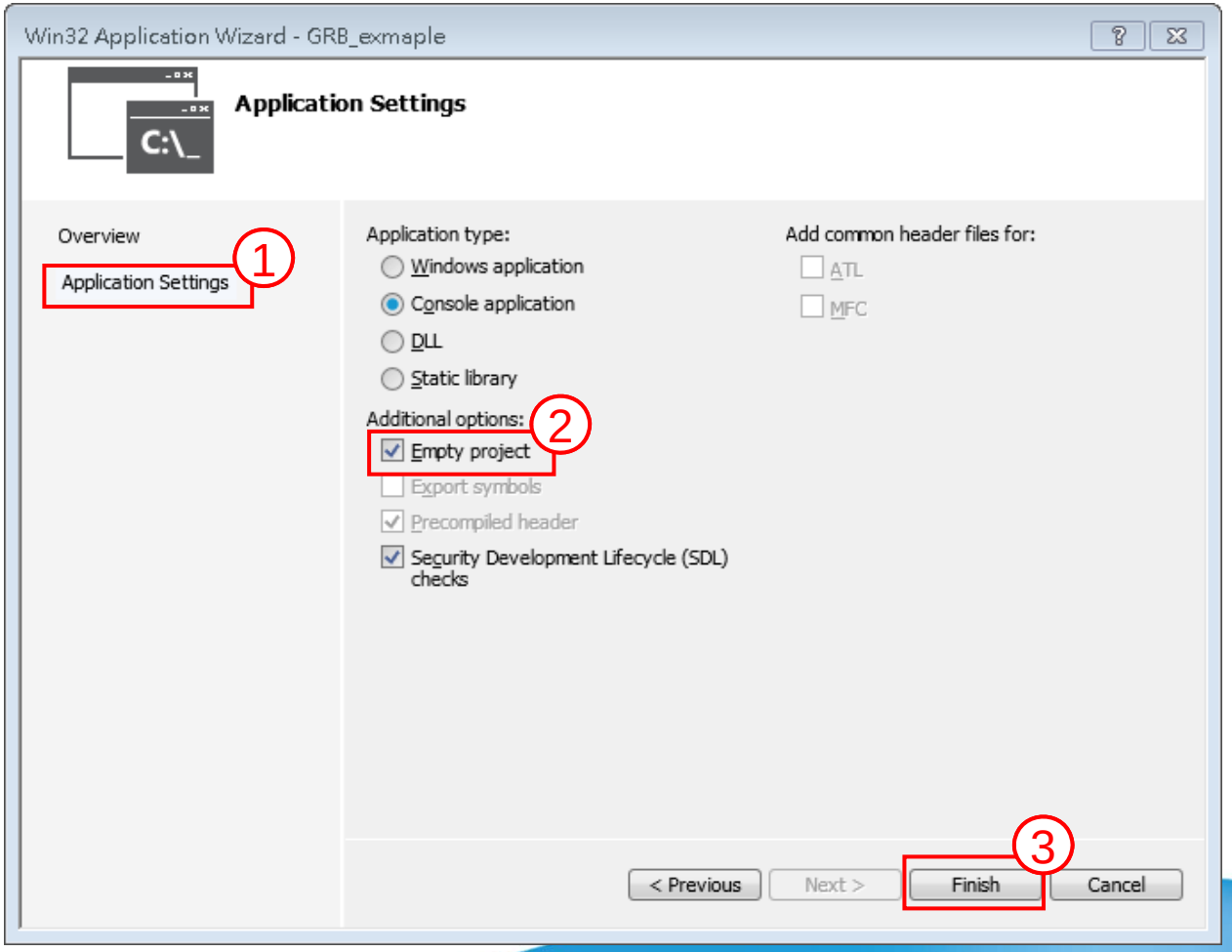
Creating Visual C++ Project (1/9)



- 『File』→『New』→『Project』
- 『Visual C++ 』→『Win32』
  →『Win32 Console Application』
- Give a name, choose a location, then click 『OK』

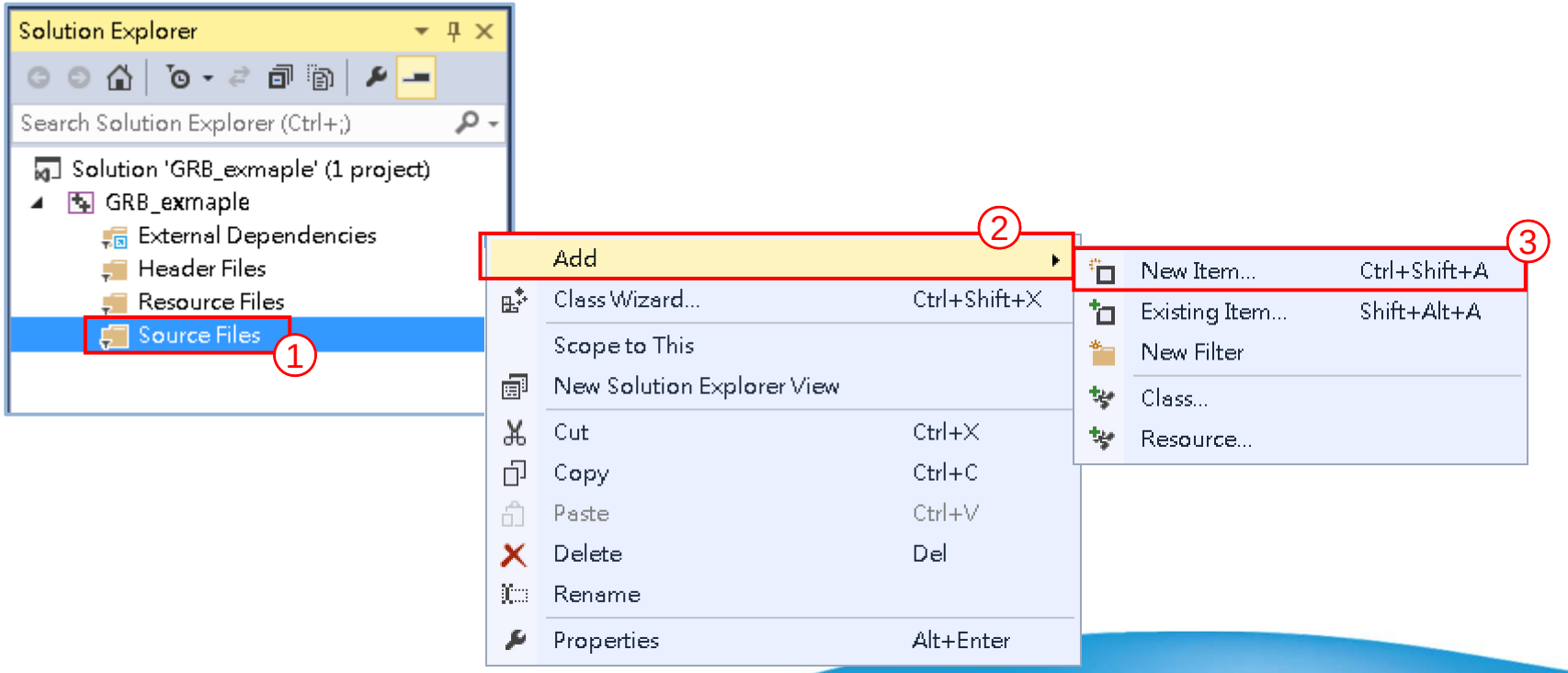


Creating Visual C++ Project (2/9)



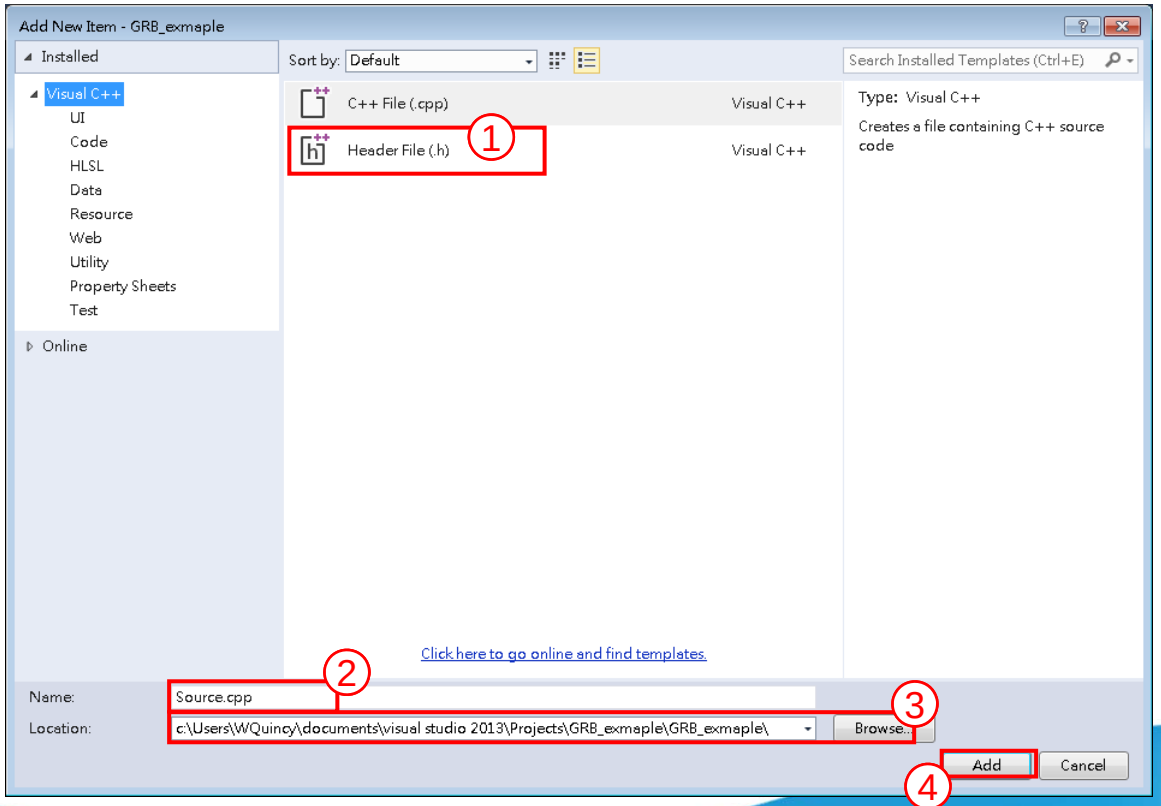
Creating Visual C++ Project (3/9)

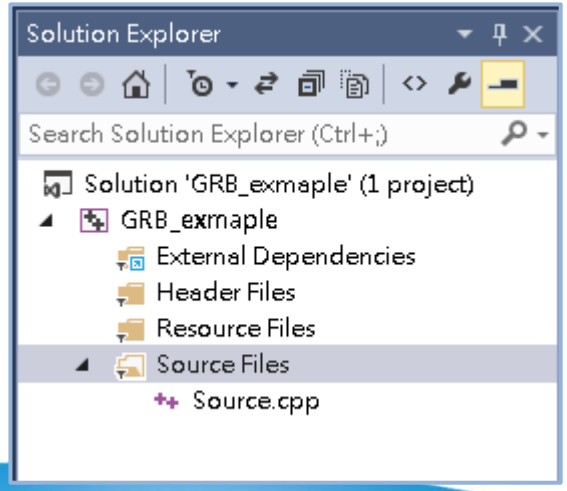
- If the CPP file exists, click 『Add』 → 『Existing Item』
- Otherwise, click 『Add』 → 『New Item』



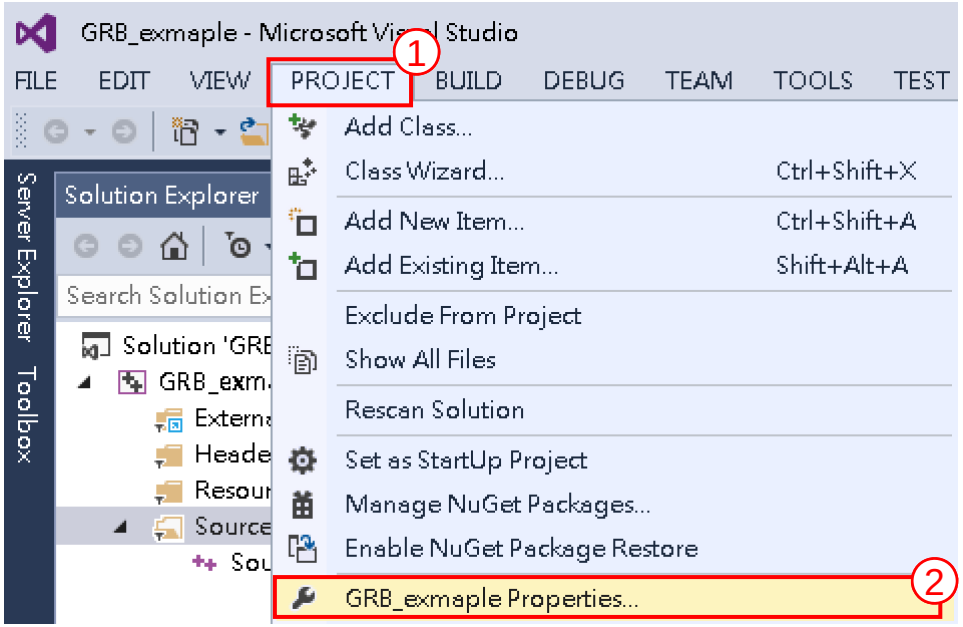
Creating Visual C++ Project (4/9)

- Choose 『C++ File (.cpp)』, fill in 『Name 』 and 『 Location』
- Then, we can find a blank cpp file shown in 『Solution Explorer』

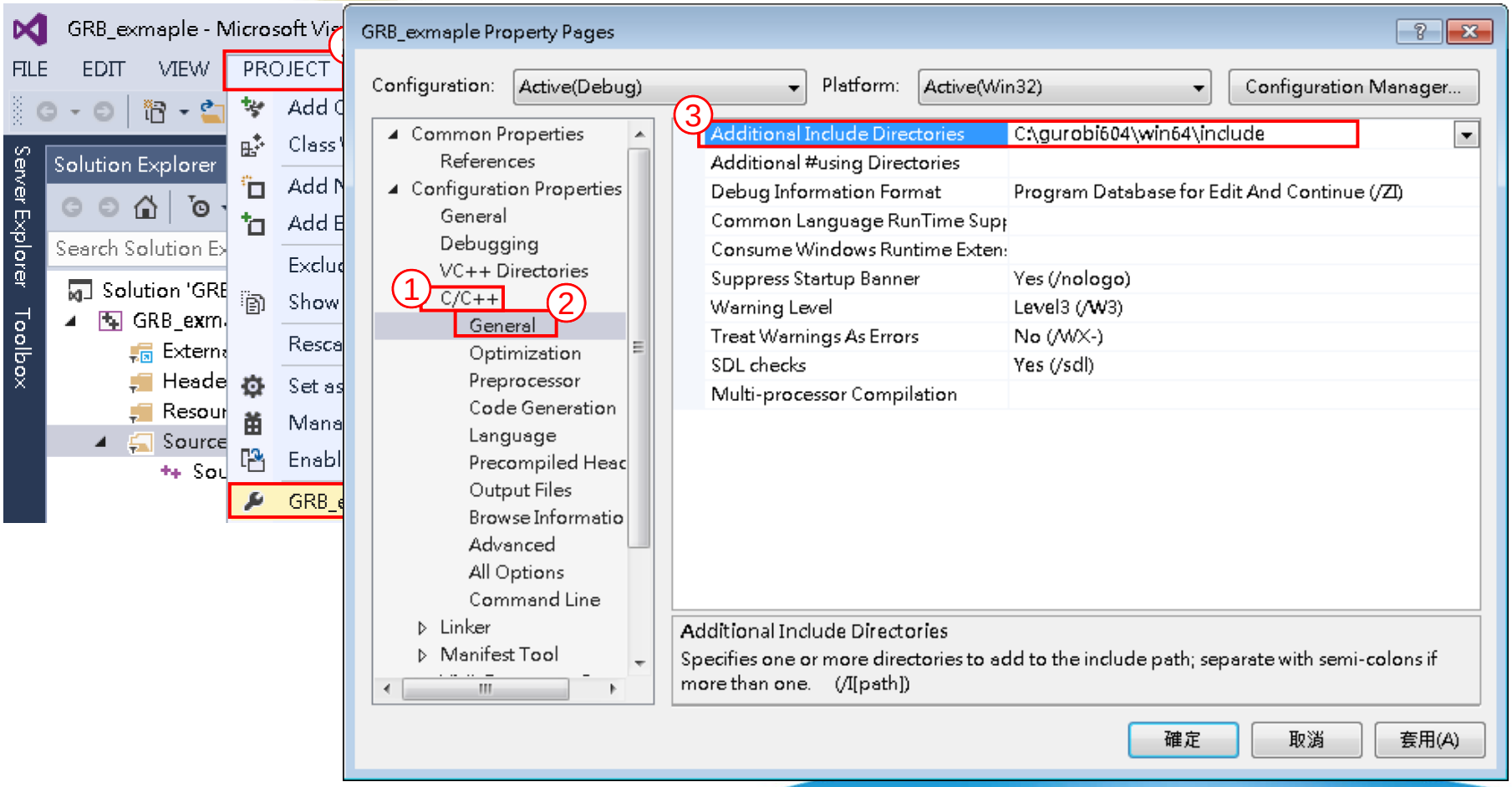




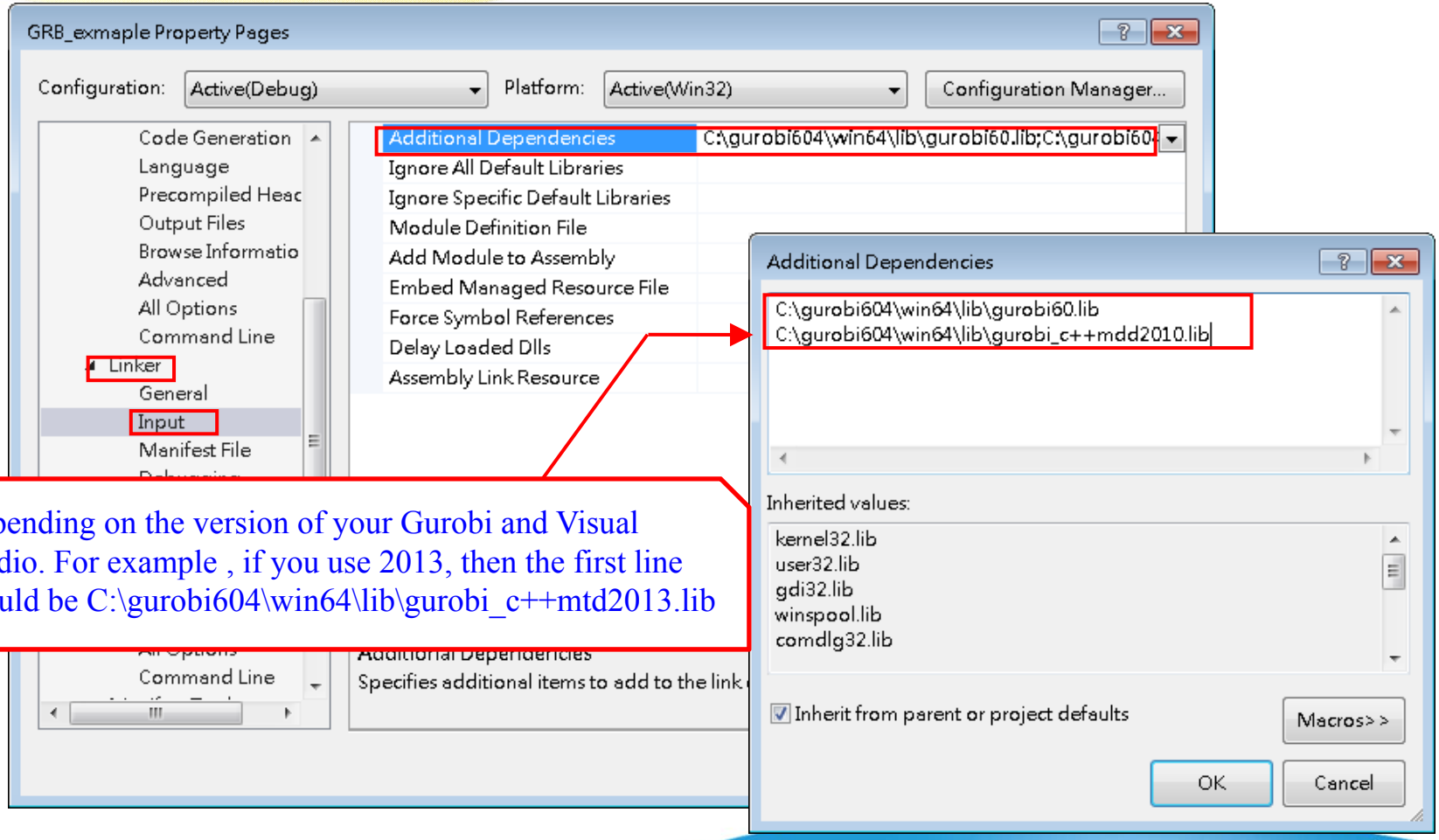
Creating Visual C++ Project (5/9)



Creating Visual C++ Project (5/9)

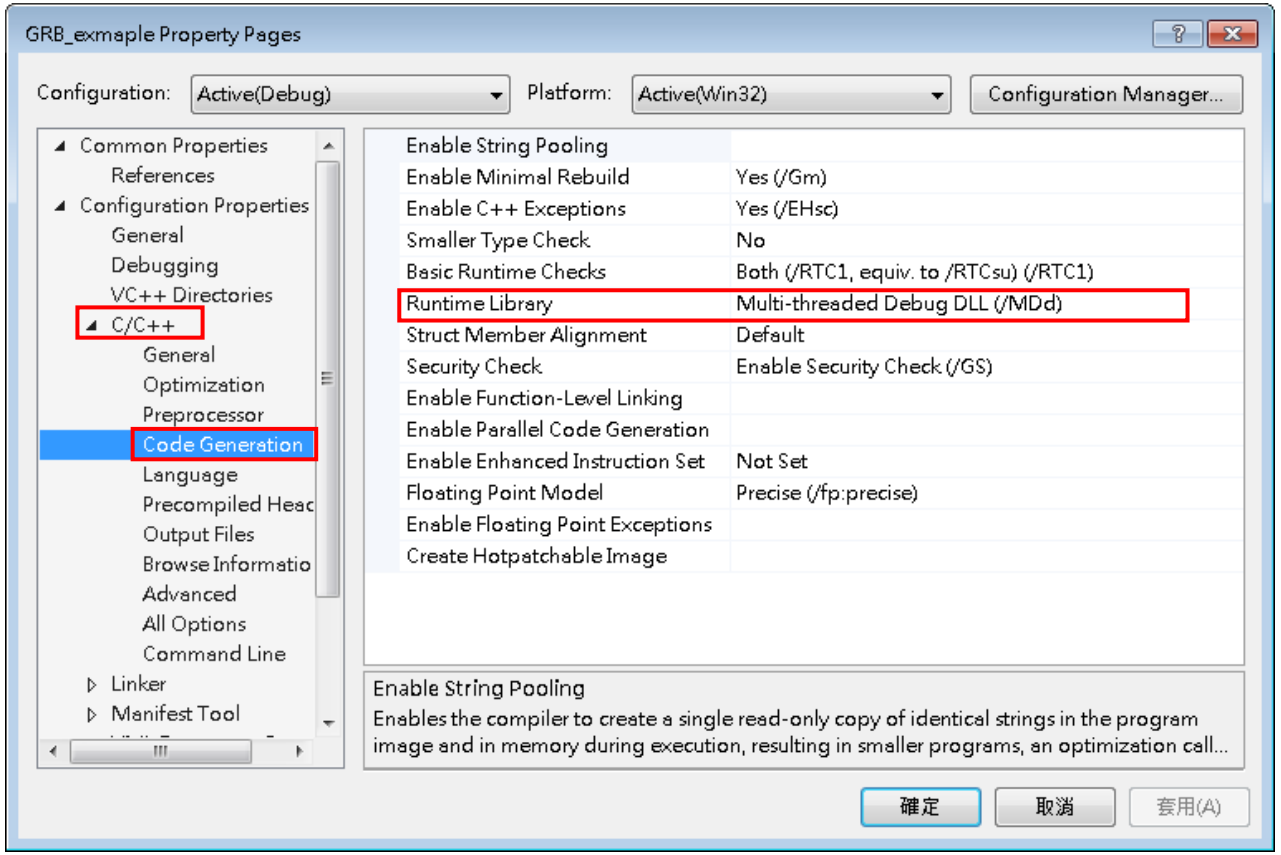


Creating Visual C++ Project (6/9)



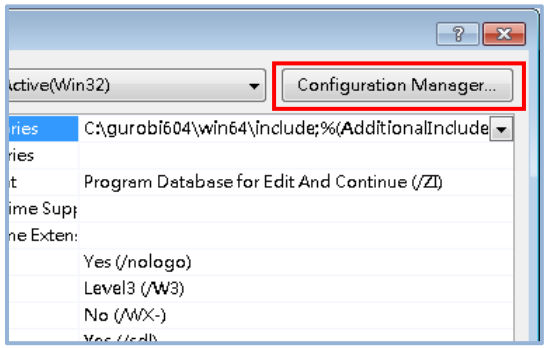
Depending on the version of your Gurobi and Visual Studio. For example , if you use 2013, then the first line should be C:\gurobi604\win64\lib\gurobi_c++mtd2013.lib

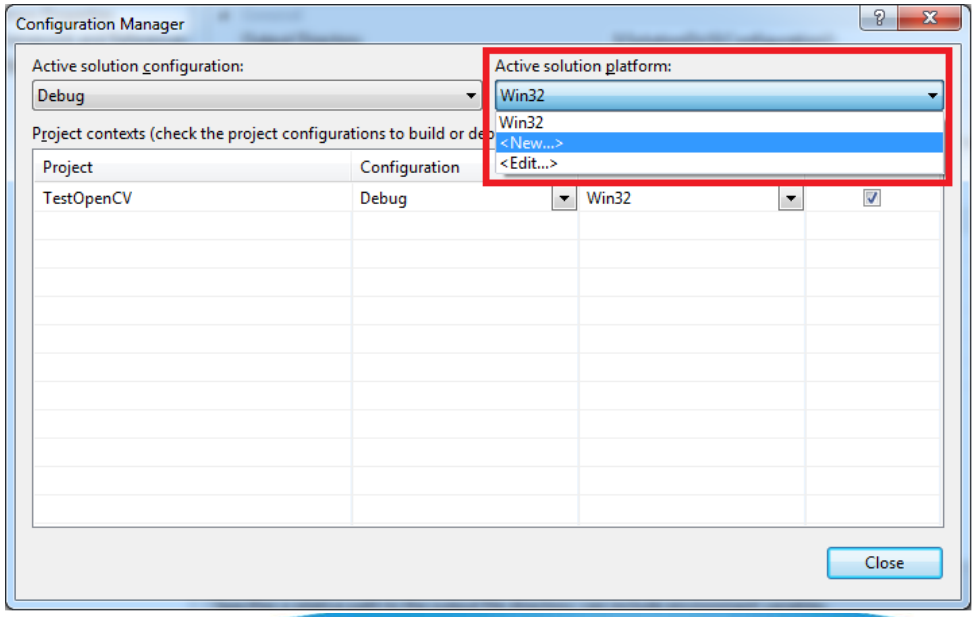
Creating Visual C++ Project (7/9)



Creating Visual C++ Project [64-bits] (8/9)

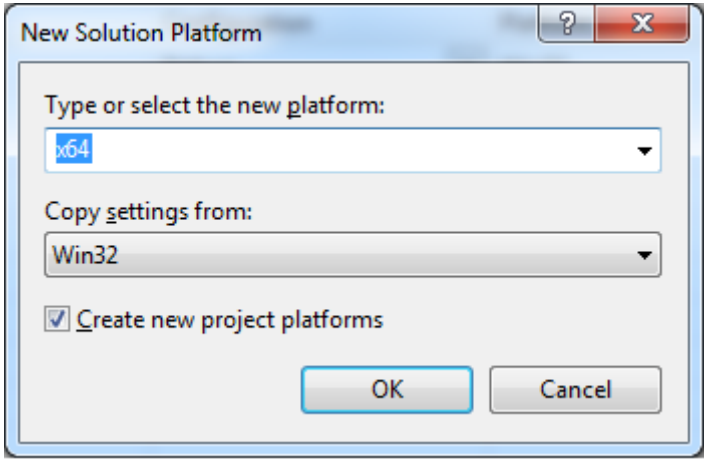
- **For 64-bits Gurobi libraries, Active solution platform has to be modified for 64-bits environment.**
- Press the Configuration Manager... button. Under Active solution platform, select New.

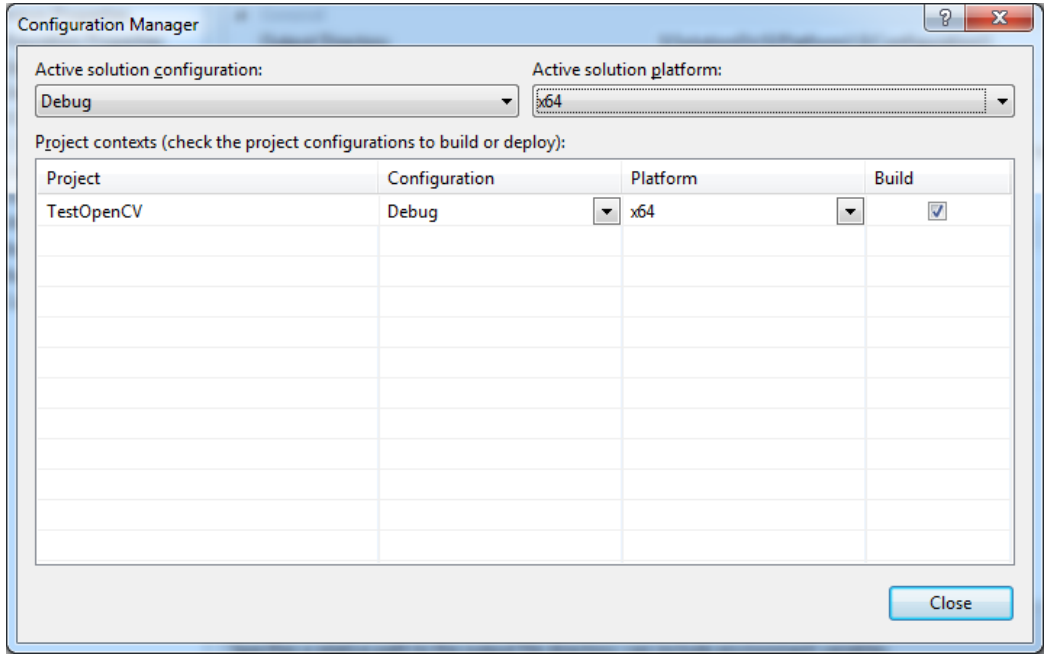




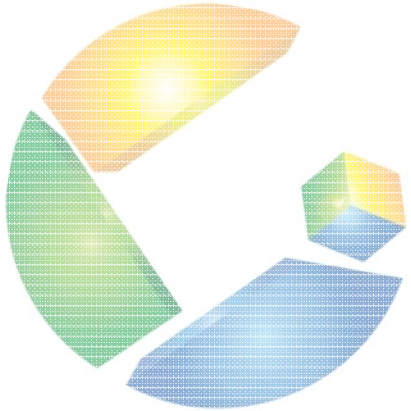
Creating Visual C++ Project [64-bits] (9/9)

- Set the new platform to x64, and press OK.





Linear Programming – Example 1



Linear Programming – Example 1

YZ Co. produces clay bowls and pots with aboriginal designs and colors. The two primary resources used by the company are skilled and special pottery clay. The two products have the following resource requirements for production and profit per item:

Product	Labor (hr/unit)	Clay (lb/unit)	Profit ($/unit)
Bowl	1	4	40
Pot	2	3	50
Available	40	120	

Product	Labor (hr/unit)	Clay (lb/unit)	Profit ($/unit)
Bowl	1	4	40
Pot	2	3	50
Available	40	120	

Decision variables：

X_1 : number of bowls to produce

X_2 : number of pots to produce

Objective function： Max $Z = 40X_1+50X_2$

Constraints：

$$X_1 + 2X_2 \leq 40$$

$$4X_1 + 3X_2 \leq 120$$

$$X_1 \geq 0$$

$$X_2 \geq 0$$

7 elements for Gurobi model

1. Basic Elements
2. Decision Variables
3. Lazy Update
4. Constraint
5. Objective Function
6. Optimization
7. Output Results

Linear Programming – model

Decision variables：

X_1 : number of bowls to produce

X_2 : number of pots to produce

Product	Labor (hr/unit)	Clay (lb/unit)	Profit ($/unit)
Bowl	1	4	40
Pot	2	3	50
Available	40	120	

Objective function： $\text{Max } Z = 40X_1 + 50X_2$

Constraints：

$$1X_1 + 2X_2 \leq 40$$
$$4X_1 + 3X_2 \leq 120$$
$$X_1 \geq 0$$
$$X_2 \geq 0$$

1.Prarmeters

2. Decition Variables

3. Constraints

4. Objective

1. Basic Elements – Gurobi Objects (1/2)

1. **Environment Object -** GRBEnv()

GRBEnv EnvName = GRBEnv();

2. **Model Object -** GRBModel (const GRBEnv& env)

GRBModel ModelName= GRBModel (EnvName);

The words in blue is the Gurobi's identifier

The words in orange is the variable identifier that can be named by yourself

```
//1.1 Basic elements declaration
GRBEnv env = GRBEnv();
GRBModel model = GRBModel(env);
```

1. Basic Elements – Parameters (2/2)

Give known parameters and coefficients by declaring a matrix or reading data from a file.

```
//1.2 Parameters definition
const int N = 2;  //number of resources
const int M = 2; //number of products (D.V.)
int a[N][M]= {{1, 2},
              {4, 3}}; //coefficients in the constraints
int b[N] = {40, 120};   //coefficients of the RHS (Right-Hand-Side)
int c[M] = {40, 50};    //coefficients of objective function
```

2. Decision Variable Declarations (1/2)

GRBVar VarName = ModelName.addVar(lb, ub, obj, type);

lb : Lower bound for the variable

ub : Upper bound for the variable (If the upper bound is unlimited, then ub is given to GRB_INFINITY)

obj : Objective coefficient for the variable.

type: GRB_INTEGER – Integer variable

GRB_BINARY – Binary variable (0 or 1)

GRB_CONTINUOUS – Continuous variable

GRB_SEMICONT – Semi-continuous variable (Ex: x=0 or 2≤x ≤4)

GRB_SEMIINT – Semi-integer variable

2. Decision Variable Declarations (2/2)

```
//2. Decision Variables
GRBVar x[M];
for(int j=0; j<M; j++) {
   x[j] = model.addVar(0.0, GRB_INFINITY, 0.0, GRB_CONTINUOUS);
}
```

The objective coefficients can be set to arbitrary value, and the true values are given later.

3. Lazy Update

Groubi update model in batch mode, so model must be updated after adding variables into the model

ModelName.update();

```
//3. Integrate variables into model
model.update();
```

4.Constraint Declaration (1/2)

GRBLinExpr Linexpr = 0;

ModelName.addConstr(lhsExpr, sense, rhsExpr);

lhsExpr : Left-hand side (LHS) expression for new linear constraint.

sense : GRB_LESS_EQUAL – LHS is less than and equal to RHS (<=).

GRB_EQUAL – LHS is equal to RHS (==).

GRB_GREATER_EQUAL – LHS is greater than and equal to RHS (>=).

rhsExpr : Right-hand side (RHS) expression for new linear constraint.

ModelName.addConstr(GRBTempConstr& tc);

4.Constraint Declaration (2/2)

Original:

$1X_1 + 2X_2 \leq 40$

$4X_1 + 3X_2 \leq 120$

General Form:

$$\sum_{j=1}^{M} a_{ij} x_j \leq b_i \qquad 1 \leq i \leq N$$

```
//4. Constraint Declaration
for(int i=0; i<N; i++) {
     GRBLinExpr LHS=0;
    for(int j=0; j<M; j++) {
          LHS += a[i][j]*x[j];
    }
    model.addConstr(LHS,GRB_LESS_EQUAL,b[i]);
}
```

5. Objective Function

ModelName.set(GRB_IntAttr_ModelSense, sense);

sense = { **1** : **Minimization** (default)
-1 : **Maximization**

ModelName.setObjective(GRBLinExpr or GRBQuadExpr**);**

GRBQuadExpr is the quadratic expression.

```
//5. set the model to maximization
model.set(GRB_IntAttr_ModelSense, -1);

GRBLinExpr Obj = 0;
for(int j=0; j<M; j++)
    Obj += c[j]*x[j];
model.setObjective(Obj);
```

Original:

$\text{Max } Z = 40X_1+50X_2$

General Form:

$$\text{Max } Z = \sum_{j=1}^{M} c_j x_j$$

6.Optimization

ModelName.optimize ();

```
//6. Optimize the model
model.optimize();
```

7. Check Optimality and Output Results

7.1 Check Optimality

Get optimality status: int status = ModelName.get(GRB_IntAttr_Status);

Status types include GRB_OPTIMAL, GRB_INF_OR_UNBD, GRB_INFEASIBLE, GRB_UNBOUNDED, etc.

```
//7.1 Check optimality
int status = model.get(GRB_IntAttr_Status);
if (status == GRB_OPTIMAL) {
        //7.2 Output the objective value and solutions
} else if (status == GRB_INF_OR_UNBD) {
        cout << "Infeasible or unbounded" << endl;
} else if (status == GRB_INFEASIBLE) {
        cout << “Infeasible" << endl;
} else if(status == GRB_UNBOUNDED) {
        cout << “Uunbounded" << endl;
} else {
        cout << "Optimization was stopped with status" << status << endl;
}
```

7. Check Optimality and Output Results

7.2 Output Results

Get objective value:

ModelName.get(GRB_DoubleAttr_ObjVal);

Get solution value:

VarName.get(GRB_DoubleAttr_X);

```
//7.2 Output the objective value and solutions
double ObjValue = model.get(GRB_DoubleAttr_ObjVal);
cout<<“total cost= "<<ObjValue<<endl;

for(int i=0; i<M; i++) {
    cout<<"x "<<i<<" =“<<x[i].get(GRB_DoubleAttr_X)<<endl;
}
```

8. Exception Handling

Using exception handling to show unexpected errors of Gurobi.

Put the following code into the main function to wrap previously mentioned steps.

```
//8 Output the objective value and solutions
int main() {
        try {
                // Step 1 to step 7 …
        } catch(GRBException e) {
                cout << "Error code = " << e.getErrorCode() << endl;
                cout << e.getMessage() << endl;
        } catch(...) {
                cout << "Exception during optimization" << endl;
        }
}
```

Comparing Different Forms

General Form

```
//2. Decision Variables
GRBVar x[M];
for(int j=0; j<M; j++)
   x[j] = model.addVar(0.0, GRB_INFINITY, 0.0,
      GRB_CONTINUOUS);
model.update();    //3. Integrate variables into model
//4. Constraint Declaration
for(int i=0; i<N; i++) {
       GRBLinExpr LHS=0;
      for(int j=0; j<N; j++) {
                LHS += a[i][j]*x[j];
      }
      model.addConstr(LHS <= b[i]);
}
//5. set the model to maximization
model.set(GRB_IntAttr_ModelSense, -1);
GRBLinExpr Obj = 0;
for(int j=0; j<M; j++)
      Obj += c[j]*x[j];
model.setObjective(Obj);
```

Expression Form

```
//2. Decision Variables
GRBVar x1, x2;
x1= model.addVar(0.0, GRB_INFINITY, 0.0,
      GRB_CONTINUOUS);
x2 = model.addVar(0.0, GRB_INFINITY, 0.0,
      GRB_CONTINUOUS);

//3. Integrate variables into model
model.update();

//4. Constraint Declaration
model.addConstr(1*x1 + 2*x2 <= 40);
model.addConstr(4*x1 + 3*x2 <= 120);

//5. set the model to maximization
model.set(GRB_IntAttr_ModelSense, -1);
model.setObjective(40*x1+50*x2);
```

Integer Programming – Example 2

Integer Programming – Example 2

A post office requires full-time employees to work on a 7days/week schedule. Every employee has to work on consecutive five days and then takes two-day off. How many employees are required for the job?

MON	TUE	WED	THU	FRI	SAT	SUN
4	5	5	10	12	12	7

Integer Programming – model

Decision variables：

X_i : the number of workers start their work on the ith day of a week，i=1,2,…,7

Objective function：

Min $z = X_1+X_2+X_3+X_4+X_5+X_6+X_7$

Constraints：

$$
\begin{aligned}
X_1 + X_4 + X_5 + X_6 + X_7 &\geq 4 \\
X_1 + X_2 + X_5 + X_6 + X_7 &\geq 5 \\
X_1 + X_2 + X_3 + X_6 + X_7 &\geq 5 \\
X_1 + X_2 + X_3 + X_4 + X_7 &\geq 10 \\
X_1 + X_2 + X_3 + X_4 + X_5 &\geq 12 \\
X_2 + X_3 + X_4 + X_5 + X_6 &\geq 12 \\
X_3 + X_4 + X_5 + X_6 + X_7 &\geq 7
\end{aligned}
$$

Integer Programming – model

Dec 2. Decition Variables

1.Prarmeters

Xi : the number of workers start their work on the ith day of a week，i=1,2,…,7

Objective function：

4. Objective

Min $z = X1+X2+X3+X4+X5+X6+X7$

3. Constraints

1.Parameters (coefficients)

$$
\begin{array}{lllllllllllll}
X_1 & & & & & X_4 & + & X_5 & + & X_6 & + & X_7 & \geq & 4 \\
X_1 & + & X_2 & & & & & X_5 & + & X_6 & + & X_7 & \geq & 5 \\
X_1 & + & X_2 & + & X_3 & & & & & X_6 & + & X_7 & \geq & 5 \\
X_1 & + & X_2 & + & X_3 & + & X_4 & & & & & X_7 & \geq & 10 \\
X_1 & + & X_2 & + & X_3 & + & X_4 & + & X_5 & & & & \geq & 12 \\
 & & X_2 & + & X_3 & + & X_4 & + & X_5 & + & X_6 & & \geq & 12 \\
 & & & & X_3 & + & X_4 & + & X_5 & + & X_6 & + & X_7 & \geq & 7
\end{array}
$$

1. Basic Elements

```
//1.1 Basic elements declaration
GRBEnv env = GRBEnv();
GRBModel model = GRBModel(env);
```

$$
\begin{array}{ccccccccccccc}
X1 & & & & & & X4 & + & X5 & + & X6 & + & X7 & \geq & 4 \\
X1 & + & X2 & & & & & & X5 & + & X6 & + & X7 & \geq & 5 \\
X1 & + & X2 & + & X3 & & & & & & X6 & + & X7 & \geq & 5 \\
X1 & + & X2 & + & X3 & + & X4 & & & & & & X7 & \geq & 10 \\
X1 & + & X2 & + & X3 & + & X4 & + & X5 & & & & & \geq & 12 \\
 & & X2 & + & X3 & + & X4 & + & X5 & + & X6 & & & \geq & 12 \\
 & & & & X3 & + & X4 & + & X5 & + & X6 & + & X7 & \geq & 7
\end{array}
$$

```
//1.2 Parameters definition
const int N = 7; //7 days per week
int a[N][N]= {{1,0,0,0,1,1,1},
              {1,1,0,0,0,1,1},
              {1,1,1,0,0,0,1},
              {1,1,1,1,0,0,0},
              {0,1,1,1,1,0,0},
              {0,0,1,1,1,1,0},
              {0,0,0,1,1,1,1}}; //coefficients for the constraints
int b[N] = {4,5,5,10,12,12,7}; //coefficients for the RHS
```

2. Decision Variable Declarations
3. Lazy Update

```
//2. Decision Variables
GRBVar x[N];
for(int i=0; i<N; i++) {
   x[i] = model.addVar(0.0, GRB_INFINITY, 0.0, GRB_INTEGER);
}
```

```
//3. Integrate variables into model
model.update();
```

4.Constraint Declaration

General Form:

$$\sum_{j=1}^{N} a_{ij} x_j \geq d_i \qquad \forall i$$

```
//4. Constraint Declaration
for(int i=0; i<N; i++) {
     GRBLinExpr LHS=0;
    for(int j=0; j<N; j++) {
          LHS += a[i][j]*x[j];
    }
    model.addConstr(LHS,GRB_GREATER_EQUAL,d[i]);
}
```

5. Objective Function
6. Optimization

```
//5. set the model to minimization
model.set(GRB_IntAttr_ModelSense,1);

GRBLinExpr Obj = 0;
for(int i=0; i<N; i++)
    Obj += x[i];
model.setObjective(Obj);
```

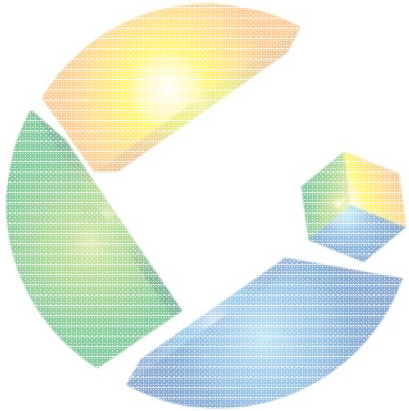
```
//6. Optimize the model
model.optimize();
```

7. Output Results

```
//7.Output the objective value and solutions
double ObjValue = model.get(GRB_DoubleAttr_ObjVal);
cout<<“total cost= "<<ObjValue<<endl;

for(int i=0; i<N; i++) {
    cout<<"x "<<i<<" =“<<x[i].get(GRB_DoubleAttr_X)<<endl;
}
```

Parameter Setting of Gurobi



Parameters (1/3) - Time limitation

GRBEnv EnvName = ModelName.getEnv();

ModelName.set(GRB_DoubleParam_ TimeLimit, **time**);

memory: Limit the total time expended (in seconds).

```
//6. Optimize the model
GRBEnv modelEnv = model.getEnv();
modelEnv.set(GRB_DoubleParam _ TimeLimit, 3600.0);
 model.optimize();
```

Parameters (2/3) - Gap

GRBEnv EnvName = ModelName.getEnv();

ModelName.set(GRB_DoubleParam_ MIPGap, **gap**);

gap : The MIP solver will terminate (with an optimal result) when the relative gap between the lower and upper objective bound is less than *MIPGap* times the upper bound

```
//6. Optimize the model
GRBEnv modelEnv = model.getEnv();
modelEnv.set(GRB_DoubleParam _ TimeLimit, 3600.0);
 model.optimize();
```

Parameters (3/3) - Reducing Memory Usage

GRBEnv EnvName = ModelName.getEnv();

EnvName.set(GRB_DoubleParam_NodefileStart, **memory**);

EnvName.set(GRB_StringParam_NodefileDir, **path**);

memory: Controls the point at which MIP tree nodes are written to disk. Whenever node storage exceeds the specified value (in GBytes), nodes are written to disk.

path: Determines the directory into which nodes are written when node memory usage exceeds the specified *NodefileStart* value.

Note: this is much more efficient than relying on virtual memory !!

```
//6. Optimize the model
GRBEnv modelEnv = model.getEnv();
modelEnv.set(GRB_DoubleParam_NodefileStart,0.1);
modelEnv.set(GRB_StringParam_NodefileDir,"G://GRBStore");
model.optimize();
```

Attributes (1/1) – Runtime and Bound

- Obtain the solving time (in seconds) for most recent optimization.

ModelName.get(GRB_DoubleAttr_Runtime);

double Runtime = model.get(GRB_DoubleAttr_Runtime);

```
//7.Output the elapsed time
double Runtime = model.get(GRB_DoubleAttr_Runtime);
cout<<“elapsed time= "<< Runtime <<endl;
```

- Obtain the best bound on current solution (lower bound for minimization, upper bound for maximization).

ModelName.get(GRB_DoubleAttr_ObjBound);

double bound = model.get(GRB_DoubleAttr_ObjBound);

```
//7.Output the lower bound (minimization problem)
double LBound = model.get(GRB_DoubleAttr_ObjBound);
cout<<“lower bound= "<< LBound <<endl;
```

More Information

Attributes for model, variable, constraints, etc:

http://www.gurobi.com/doc/40/refman/node571.html#sec:Attributes

Parameters for solving scheme:

http://www.gurobi.com/doc/40/refman/node572.html#sec:Parameters

Status codes for optimization:

http://www.gurobi.com/doc/40/refman/node576.html#sec:StatusCodes