Multiple Files Compilation

If We Want to Write a Program with our Linked List

```
int main()
{
    List l;

    l.insertHead(123);
    l.insertHead(11);
    l.insertHead(9);
    l.insertHead(1);
    l.insertHead(20);

    for (int i = 0; i < 5; i++) {
        cout << "The current list is: ";
        l.print();
    }

    return 0;

}
```

Where should we put this?

One Big .cpp File?

Code for Linked List

```
class ListNode {
private:
  int item;
  ListNode *next;
…
}
….
```

Code for the main function

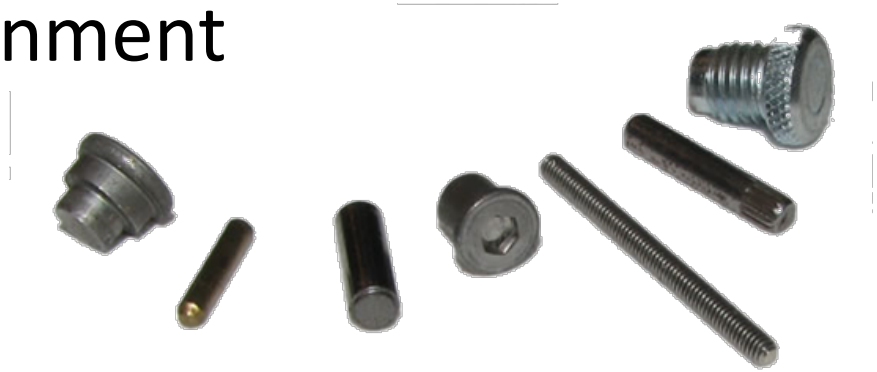
```
int main() {


}
```

If We Lump Every Code into ONE SINGLE File

- Microsoft Windows operating system has roughly **50 million lines** of code.
- The file is too large to
  - load/save
  - be understood
  - search for errors
- One single file cannot be shared/distributed if you have more than one programmer

Project Sizes in University

- Course assignment



- Course project/FYP

Project Size at Work

Breaking Into Multiple Files

- Logically, we can break our code into various files based on their functionality

LinkedList.cpp

Code for Linked List

```
class ListNode {
private:
  int item;
  ListNode *next;
…
}
….
```

main.cpp

Code for the main function

```
int main() {


}
```

However

- Then the main.cpp will be

```
int main()
{
    List l;

    l.insertHead(123);
    l.insertHead(11);
    l.insertHead(9);
    l.insertHead(1);
    l.insertHead(20);

    for (int i = 0; i < 5; i++) {
        cout << "The current list is: ";
        l.print();
    }

    return 0;

}
```

main.cpp

Compilation ERROR!!!

Because “List” is not declared

- We can put the declaration of “List” into main.cpp without the body or implementation of “List”
- But then we have to copy the declaration to every file if “List” is used?

```
class ListNode
{
private:
    int _item;
    ListNode *_next;

public:
    ListNode(int);
    int content() { return _item; };
    friend class List;
};
```

```
int main()
{
    List l;

    l.insertHead(123);
    l.insertHead(11);
    l.insertHead(9);
    l.insertHead(1);
    l.insertHead(20);

    for (int i = 0; i < 5; i++) {
        cout << "The current list is: ";
        l.print();
    }

    return 0;

}
```

main.cpp

Header File

- We separate the code for Linked List into two files
  - ".h file", the declaration of all classes and functions
  - ".cpp file", function bodies and implementations

LinkedList.h

```
class ListNode
{
private:
    int _item;
    ListNode *_next;
public:
    ListNode(int);
  // etc. et.c
};

class List
{
private:
    int _size;
    ListNode *_head;

public:
    List()
     ~List();
    void insertHead(int);
  // etc. etc.
};
```

LinkedList.cpp

```
ListNode::ListNode(int n)
{
    _item = n;
    _next = NULL;
}

void List::insertHead(int n)
{
    ListNode *aNewNode
       = new ListNode(n);
    aNewNode->_next = _head;
    _head = aNewNode;
    _size++;
};

// etc. etc….
```

#include

- But then
  - Compilation error because no declaration of List/ListNode
- Use #include to “paste” the whole file of “LinkedList.h” into the .cpp file

LinkedList.cpp

```
#include “LinkedList.h”

ListNode::ListNode(int n)
{
    _item = n;
    _next = NULL;
}

void List::insertHead(int n)
{
    ListNode *aNewNode
       = new ListNode(n);
    aNewNode->_next = _head;
    _head = aNewNode;
    _size++;
};

// etc. etc….
```

However, What if?

file1.h

```
class Whatever {
.
.
}
```

file2.h

```
#include “file1.h”
```

file3.h

```
#include “file1.h”
#include “file2.h”
```

file1.cpp

```
#include “file1.h”
```

file2.cpp

```
#include “file2.h”
```

file3.cpp

```
#include “file3.h”
```

Included “file1.h” more than one time?!
Cause ERROR because class Whatever is declared twice here

#pragma once

file1.h

```
#pragma once

class Whatever {
.
}
```

file1.cpp

```
#include “file1.h”
```

- use “#pragma once” to make sure the file will appear only once even it is included a few times

Alan’s Ph.D Code

```
hcheng@suna0:~/softwares/Skin/Skin back 3-26[1031]$ ls -l
total 72
drwx------   2 hcheng   compsc      4096 Mar 26  2003 basic
drwx------   2 hcheng   compsc      4096 Mar 26  2003 CompDB
drwx------   2 hcheng   compsc      4096 Mar 26  2003 delone
drwx------   2 hcheng   compsc      4096 Mar 26  2003 geometry
drwx------   2 hcheng   compsc      4096 Mar 26  2003 li
drwx------   3 hcheng   compsc      4096 Mar 26  2003 Skin
drwx------   3 hcheng   compsc      4096 Mar 26  2003 Skin.bak
drwx------   2 hcheng   compsc      4096 Mar 26  2003 SkinMesh
drwx------   2 hcheng   compsc      4096 Mar 26  2003 sos
```

```
basic:
arg.c            data.cpp          rarray.h              rgitypes.h        versions.h
basic.c          data.h            rarray.hpp            rgivector.cpp     vertexarray.cpp
basic.dsp        ex.h              rectsel.cpp           rgivector.h       vertexarray.h
basic.h          ex.hpp            rectsel.h             rgivector.hpp     vltdata.cpp
basic.plg        .h                rgicstring.cpp        spectrum.cpp      vltdata.h
binio.cpp        .hpp              rgicstring.h          spectrum.h        wfshortestpather.cpp
binio.h          imer.cpp          rgicstring.hpp        stackbv.cpp       wfshortestpather.h
binio.hpp        imer.h            rgimatrix.cpp         stackbv.h         win_basic.h
bitvector.       s.cpp             rgimatrix.h           stringtable.cpp   wireframe.cpp
bitvector.       s.h               rgimatrix.hpp         stringtable.h     wireframe.h
bitvector.hpp       convert.hpp     history.h        malloc.c        points.hpp      rgimessage.cpp        time.c            wireframe.hpp
build.h             data.cpp        iit.c            map.h           pqueue.h        rgimessage.h          tokenize.c        xdr.c
callbacklist.cpp    data.h          index.h          map.hpp         pqueue.hpp      rgimessagestack.cpp   tritype.h         xdr.h
callbacklist.h      dumpable.cpp    isort.c          math2.c         prime.c         rgimessagestack.h     uf.c
callbackobject.cpp  dumpable.h      iterstack.h      miscmath.cpp    qsort.c         rgistring.cpp         unix_basic.h
callbackobject.h    dumpable.hpp    iterstack.hpp    miscmath.h      queue.h         rgistring.h           util.h
cb_doprnt.c         facepoint.h     kdtree.cpp       multitree.h     queue.hpp       rgitranslator.cpp     vectmat.cpp
cb.c                farray.h        kdtree.h         multitree.hpp   raindrop.c      rgitranslator.h       vectmat.h

CompDB:
compDB.cpp  CompDB.dsp  compDB.h   CompDB.plg

delone:
boundary.cpp  dcbuilder.cpp  dcofaces.cpp  dcomp.cpp   dcompiter.cpp  delone.dsp   faces.cpp      ksimpsize.h   simpsize.h
boundary.h    dcbuilder.h    dcofaces.h    dcomp.h     dcompiter.h    delone.plg   ksimpsize.cpp  simpsize.cpp

geometry:
animate.cpp   comp.cpp          edgeset.h         ihandler.cpp   modtrinfo.hpp        segmenttree.cpp       simplex.h       trist.cpp            vertarray.h
animate.h     comp.h            fliphandler.cpp   ihandler.h     orienter.cpp         segmenttree.h         simplexset.cpp  trist.h              vertex.cpp
boxes.cpp     computil.cpp      fliphandler.h     ksimplex.cpp   orienter.h           shortestpather.cpp    simplexset.h    trist.hpp            vertex.h
boxes.h       computil.h        geometry.dsp      ksimplex.h     ortribv.cpp          shortestpather.h      testint.cpp     tristconnector.cpp   vertset.cpp
bvtag.cpp     edgecycleset.cpp  geometry.plg      locate.cpp     ortribv.h            simph.cpp             testint.h       tristconnector.h     vertset.h
bvtag.h       edgecycleset.h    geomutil.cpp      locate.h       packedihandler.cpp   simph.h               tolerancer.cpp  tristmodifier.h
cofaces.h     edgeset.cpp       geomutil.h        modtrinfo.h    packedihandler.h     simplex.cpp           tolerancer.h    vertarray.cpp

li:
base.h    det.c    li.dsp   li.hpp   lia.c   liaux.c      lidet.cpp   liminor.cpp   lipoints.cpp   listack.cpp   pool.c
chars.c   li.cpp   li.h     li.plg   lia.h   liaux.c.old  lidet.h     liminor.h     lipoints.h     listack.h     stack.c

Skin:
a.h                    ChildFrm.cpp        FormCommandView.cpp   MainFrm.h         resource.h   Skin.dsw   Skin.plg      SkinView.cpp
AlphaDlg.cpp           ChildFrm.h          FormCommandView.h     ReadMe.txt        Skin.aps     Skin.h     Skin.rc       SkinView.h
AlphaDlg.h             dump.stl            InputCQ.cpp           RenderView.cpp    Skin.clw     skin.log   Skin.reg      StdAfx.cpp
beforeRefinement.sav   FileOpenOption.cpp  InputCQ.h             RenderView.h      Skin.cpp     Skin.ncb   SkinDoc.cpp   StdAfx.h
beforeRefinement.stl   FileOpenOption.h    MainFrm.cpp           res               Skin.dsp     Skin.opt   SkinDoc.h
```

How to Compile Multiple Files?

- E.g. we have
  - LinkedList.h
  - LinkedList.cpp
  - main.cpp
- in **VSCode**, you can simply change to the directory contains the files in the terminal and type:

```
g++ LinkedList.cpp main.cpp
```

- Noted that you don't need to add in ".h"

Exectuables

- If there is not error, the executable will be “a.exe” in the same directory. Just type “a.exe” to run the program
- You can rename “a.exe” to another name or simply give it a name, e.g. “myProg.exe” when it compiles by option “-o”

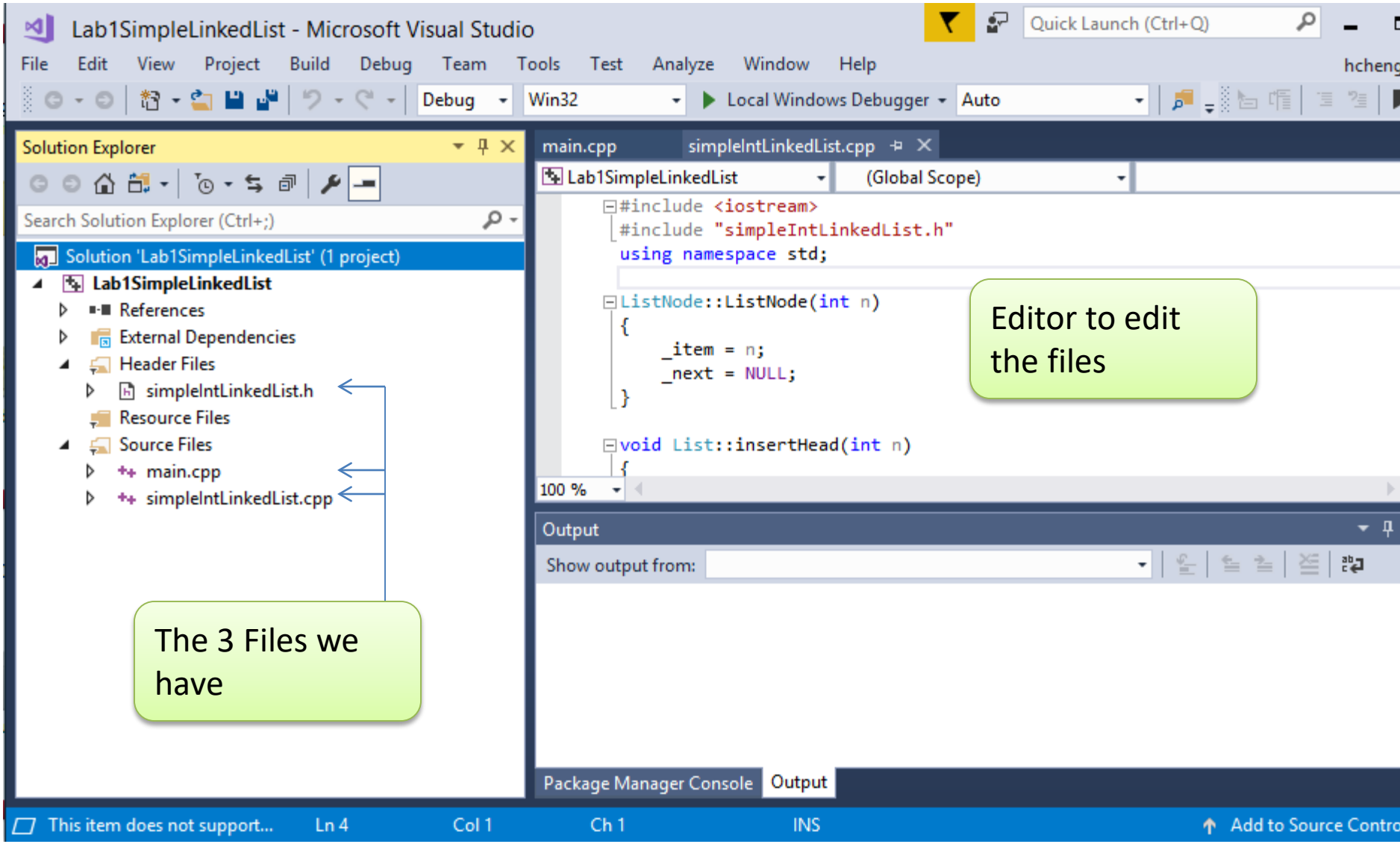
```
g++ LinkedList.cpp main.cpp -o myProg.exe
```

Using MS Studio

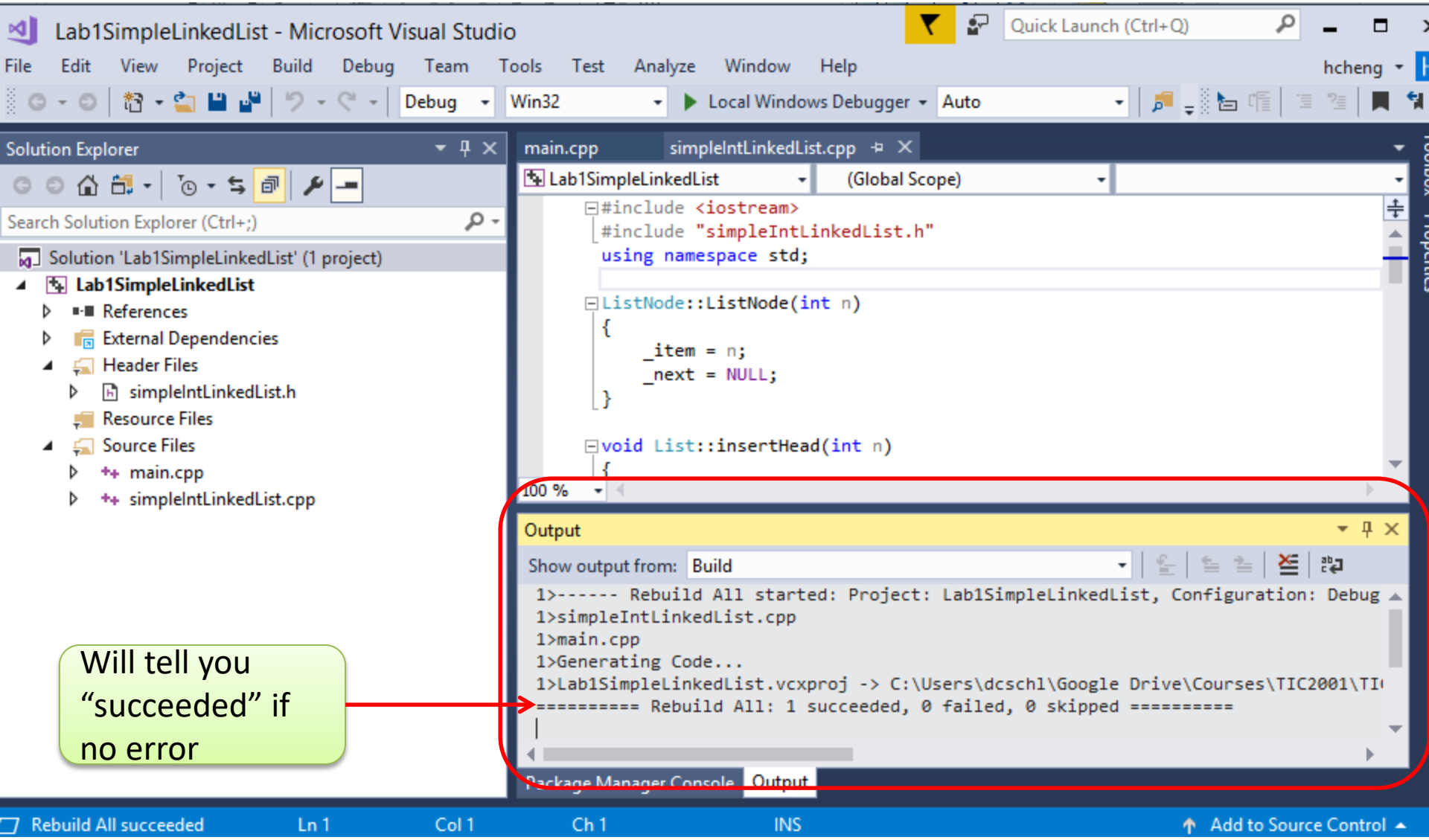
- In MSVS C++ Studio, you can create a "**solution**"(project) to compile multiple files
- But for our assignments, we will created the .sln file for you

Try Our Assignment One

- Download assignment from coursemology and unzip it
- Find the .sln file inside and double click it


Lab1SimpleLinkedList - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
hcheng
Debug
Win32
Local Windows Debugger
Auto
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Lab1SimpleLinkedList' (1 project)
Lab1SimpleLinkedList
References
External Dependencies
Header Files
simpleIntLinkedList.h
Resource Files
Source Files
main.cpp
simpleIntLinkedList.cpp
The 3 Files we have
main.cpp
simpleIntLinkedList.cpp
Lab1SimpleLinkedList
(Global Scope)
#include <iostream>
#include "simpleIntLinkedList.h"
using namespace std;
ListNode::ListNode(int n)
{
_item = n;
_next = NULL;
}
void List::insertHead(int n)
{
Editor to edit the files
100 %
Output
Show output from:
Package Manager Console
Output
This item does not support...
Ln 4
Col 1
Ch 1
INS
Add to Source Contro

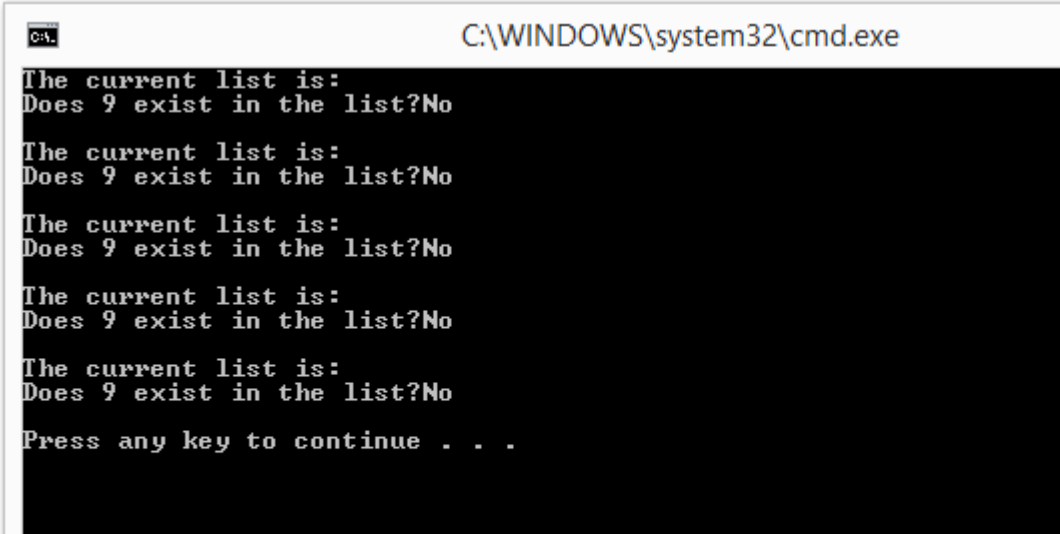
If You Compile (Build) by “F7”



Will tell you “succeeded” if no error

Compile and Run

- To Compile your code
  - Build > Build Solution
- To run your code
  - Debug > Start Without Debugging
  - Or simply press “ctrl-F5”
- Example Output:



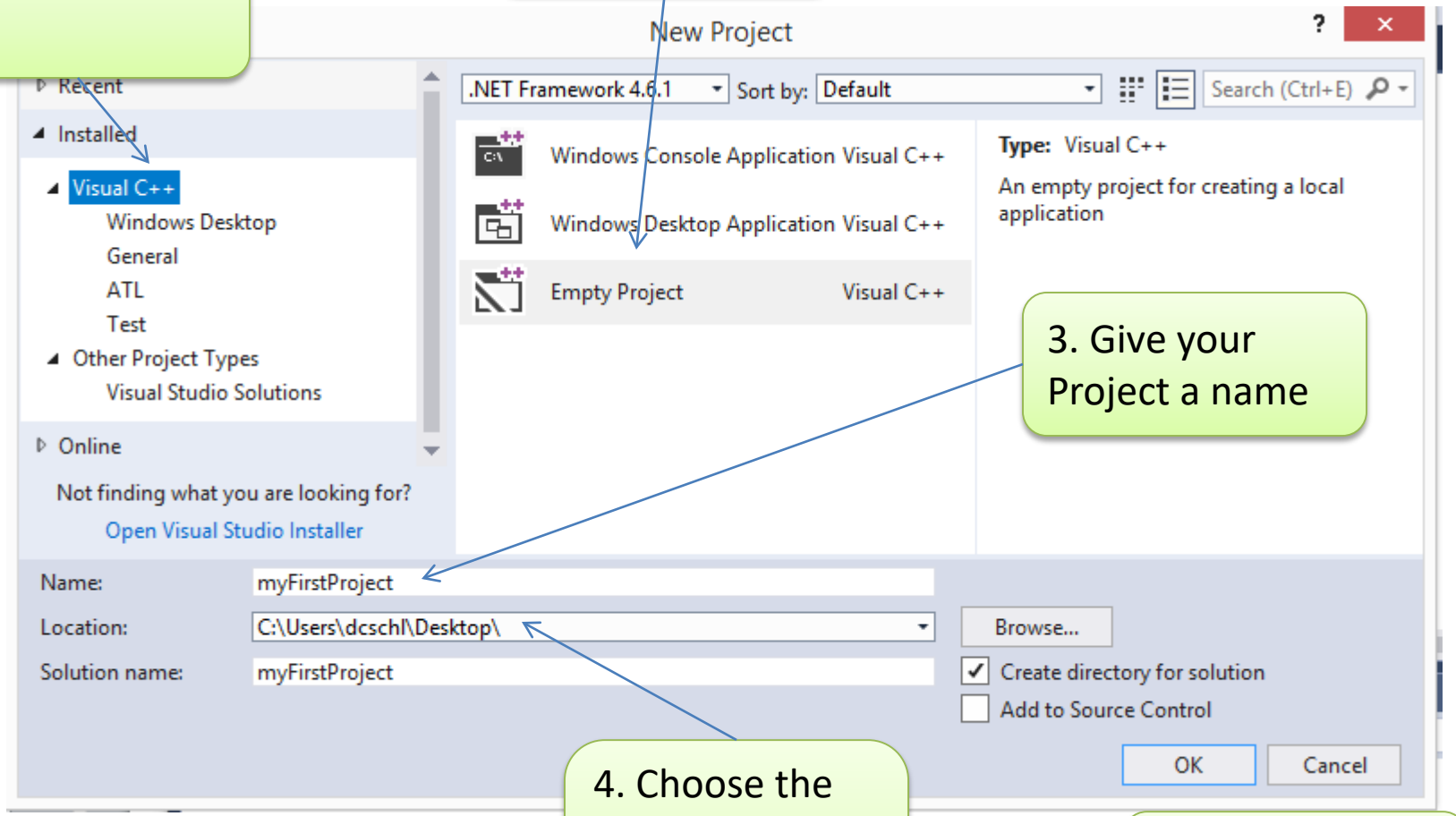
Creating a “Solution” in MSVS

- But if you want to create your own project
  - In which you shouldn’t need to do it for our assignments
- You can create a solution by
  - File > New > Project
  - Or simply “Ctrl-shift-N”

To Crea[illegible]mple WS

1. Click Visual C++

2. Select “Empty Project”



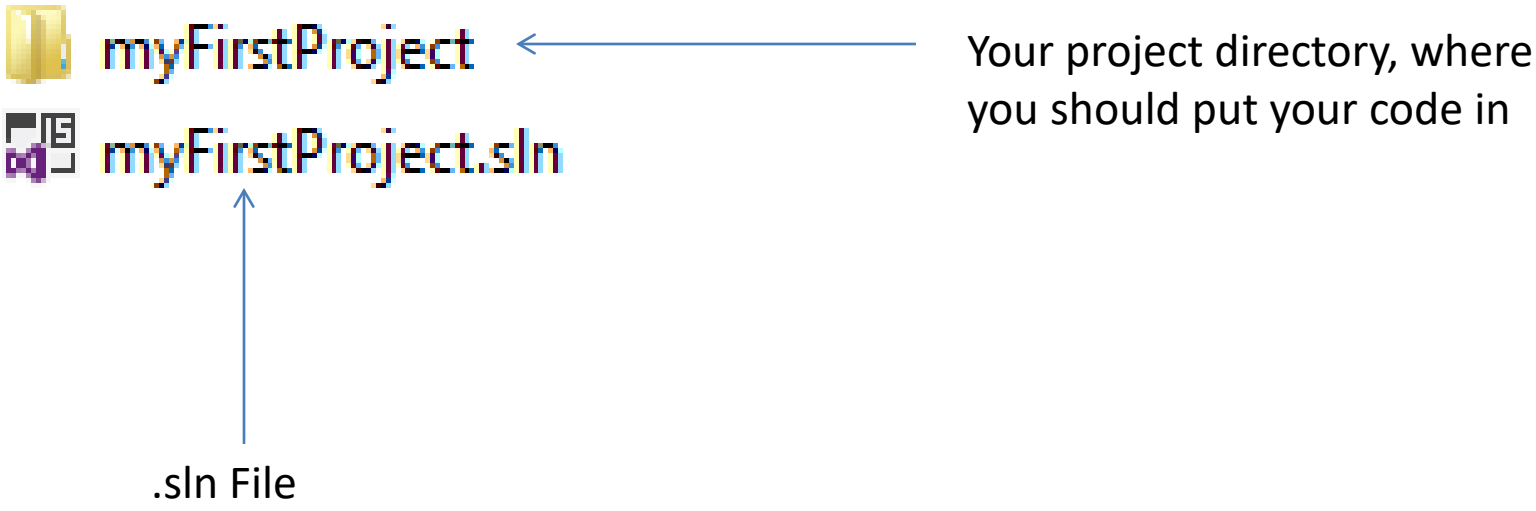
3. Give your Project a name

4. Choose the directory you want to place the folder

5. Click “OK”

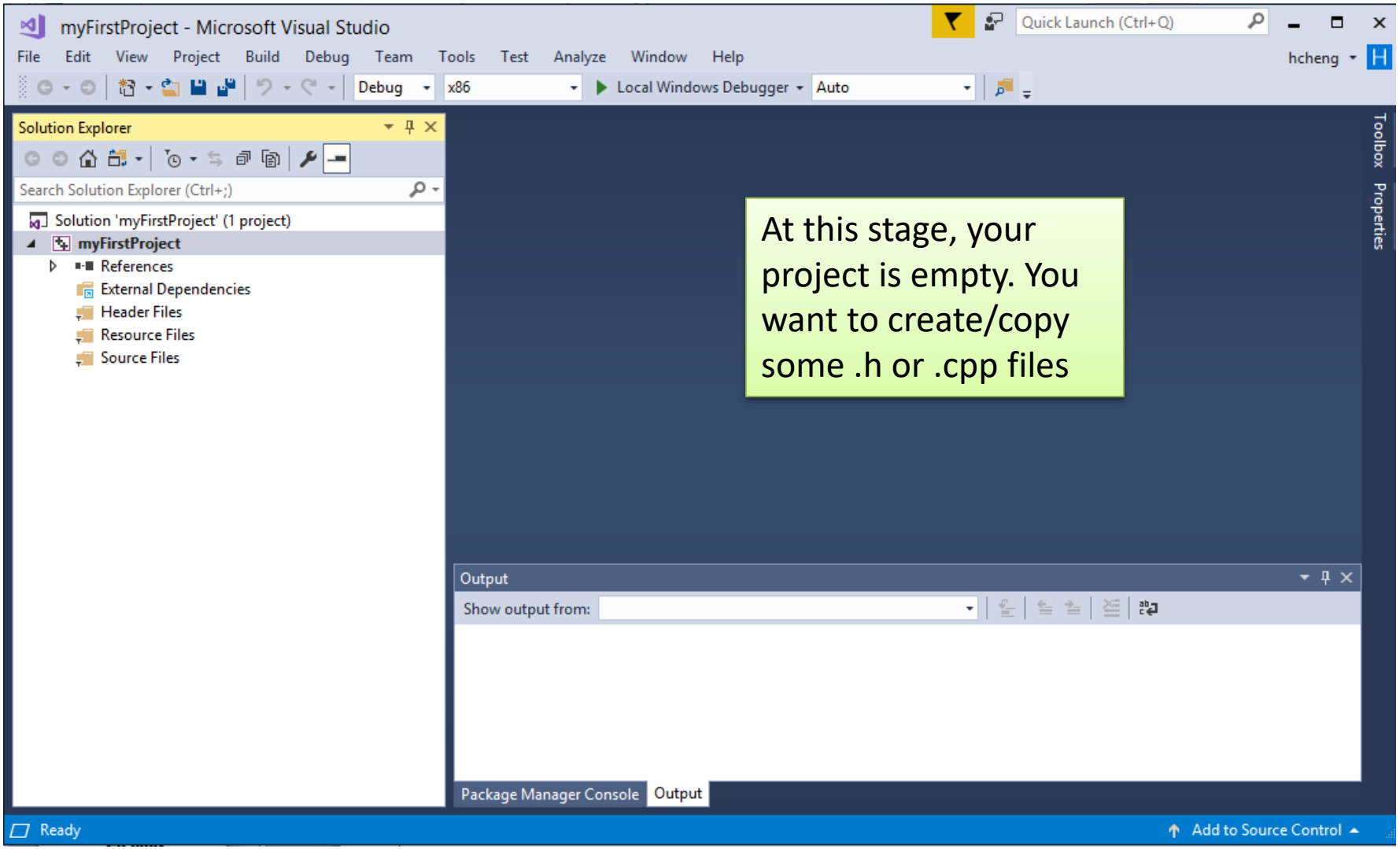
Files created

- In the folder you created, you will find:



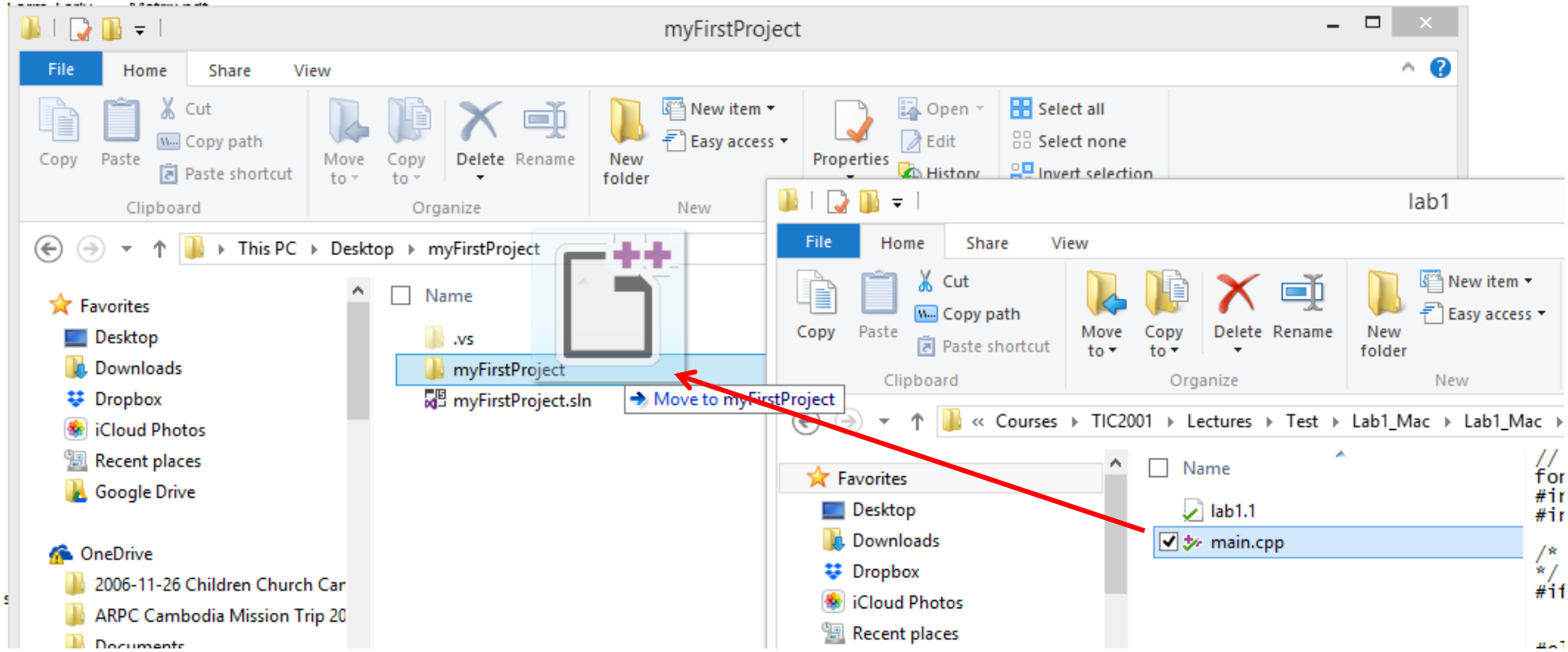
- Whenever you want to re-open your project, just click the .sln file

Empty Project



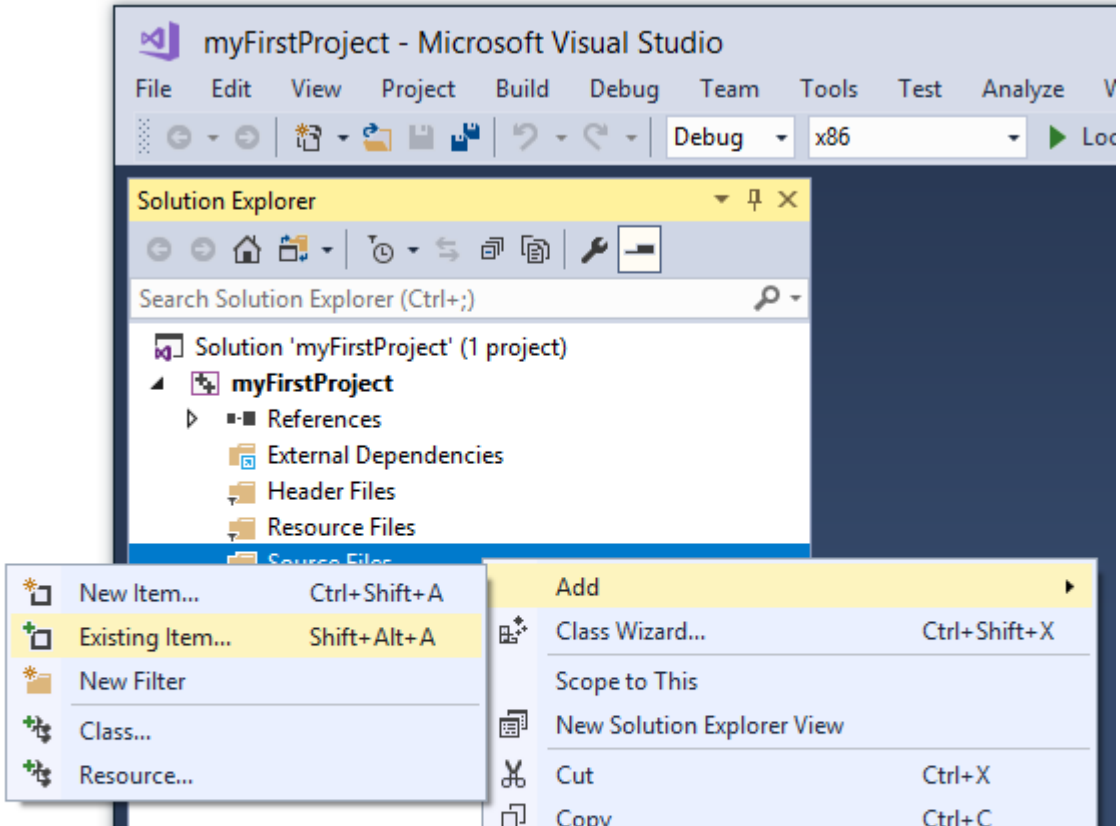
Copy Old .cpp File

- If you already have some .cpp file (e.g. from your prev. course) you want to compile
  - Copy the .cpp into your project directory



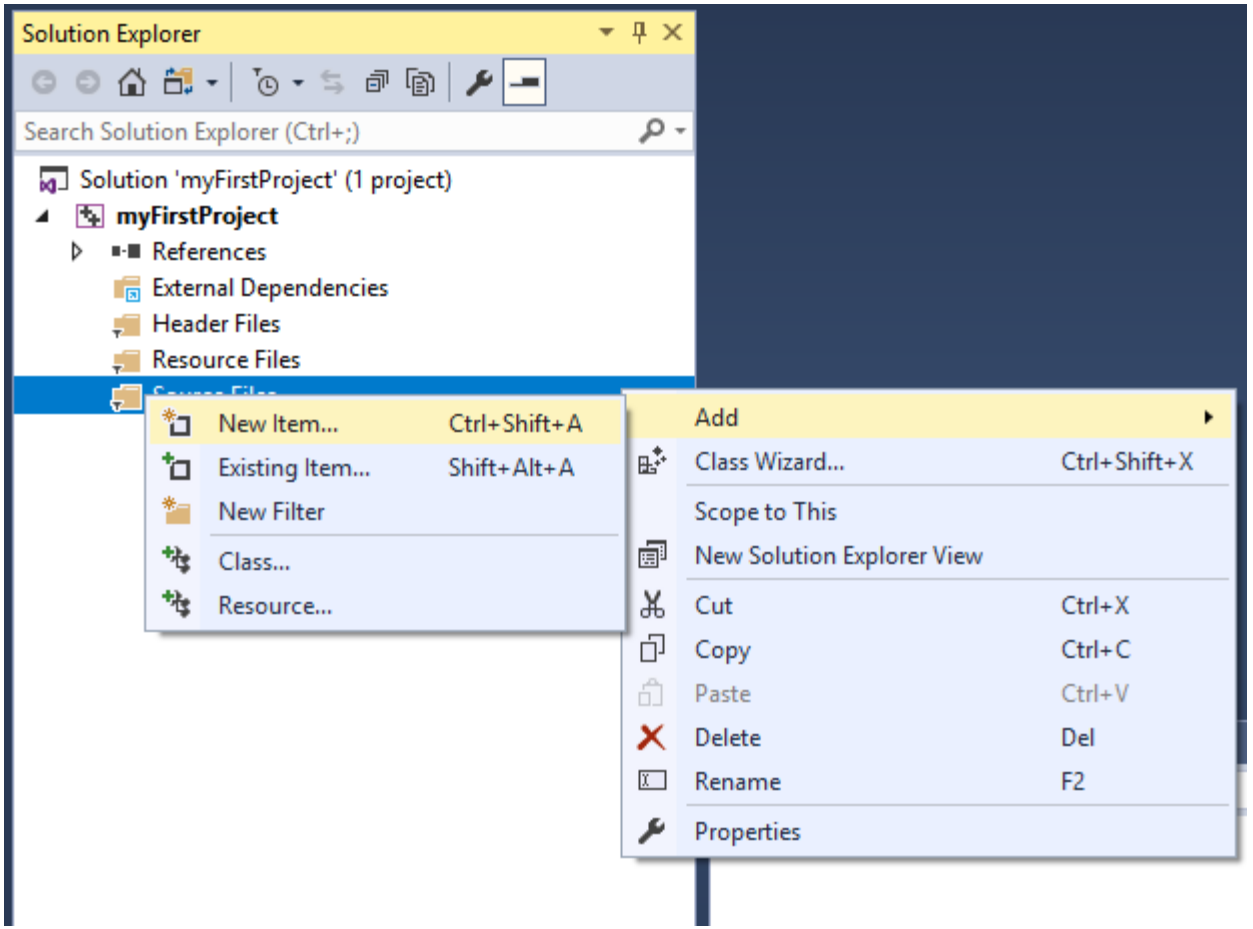
Then in Your Project

- Right Click “Source Files” > Add > Existing Item
- Then find the file you just copied

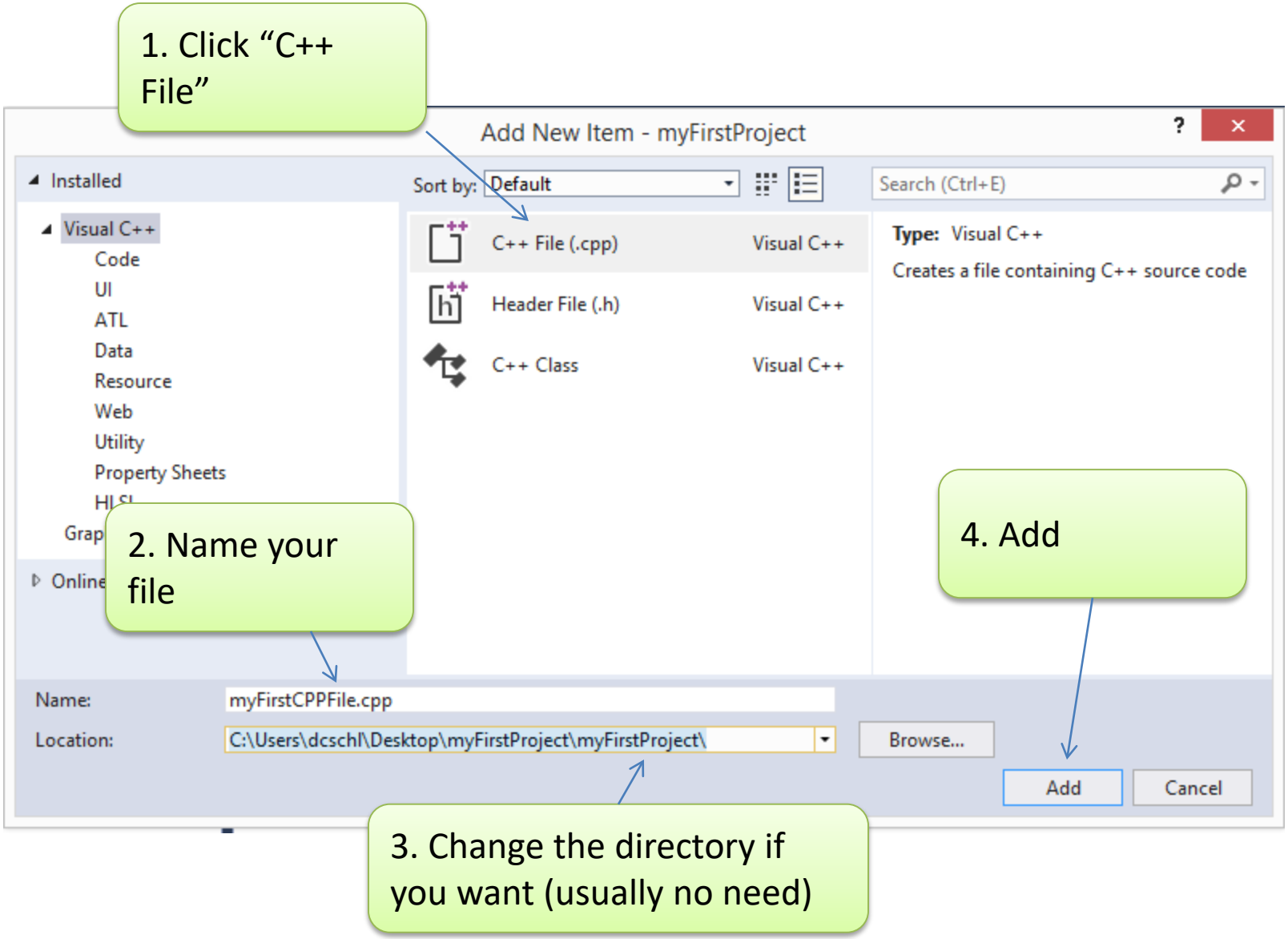


Or, To Create a .cpp File from Scratch

- Right click “Source Files” > Add > New Item



Or, To Create a .cpp File from Scratch



Compile and Run (Same same)

- To Compile your code
  - Build > Build Solution
- To run your code
  - Debug > Start Debugging
  - Or simply press “F5”

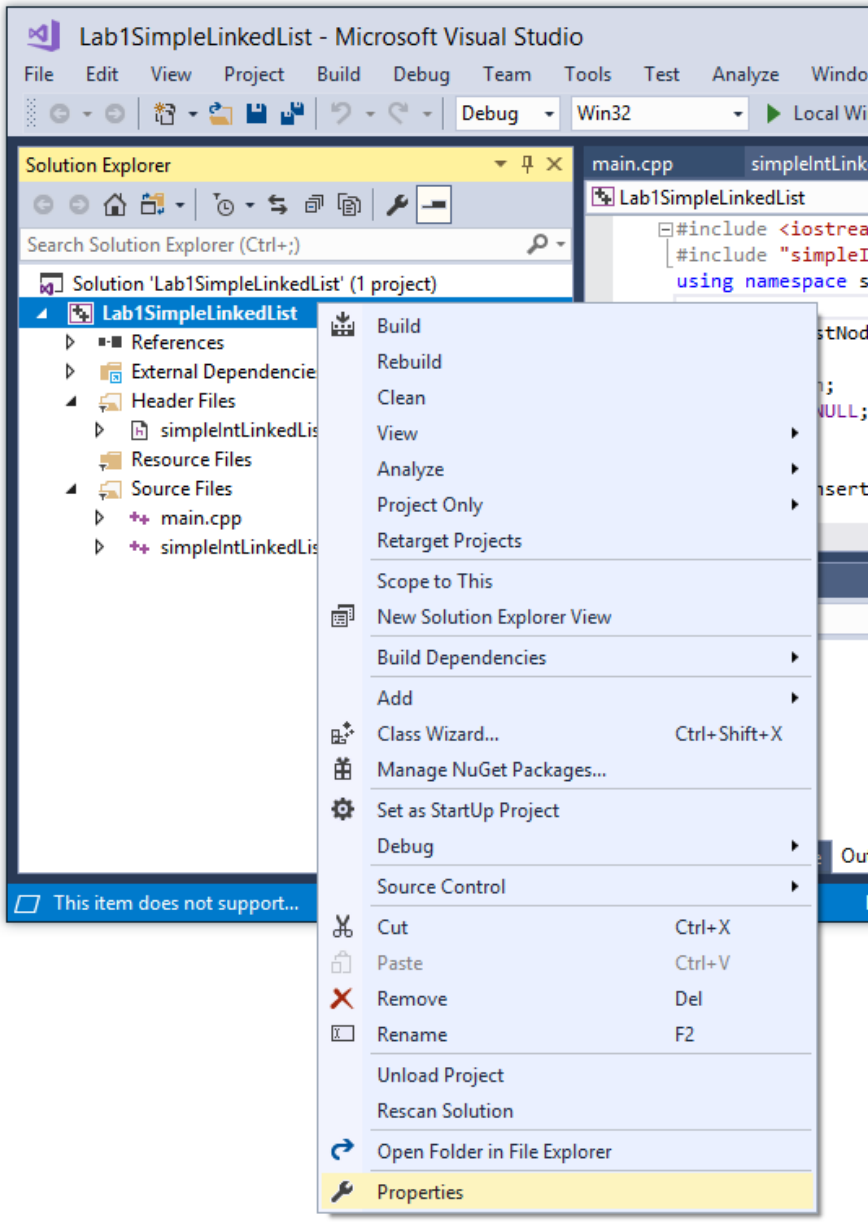
If You Run the Program

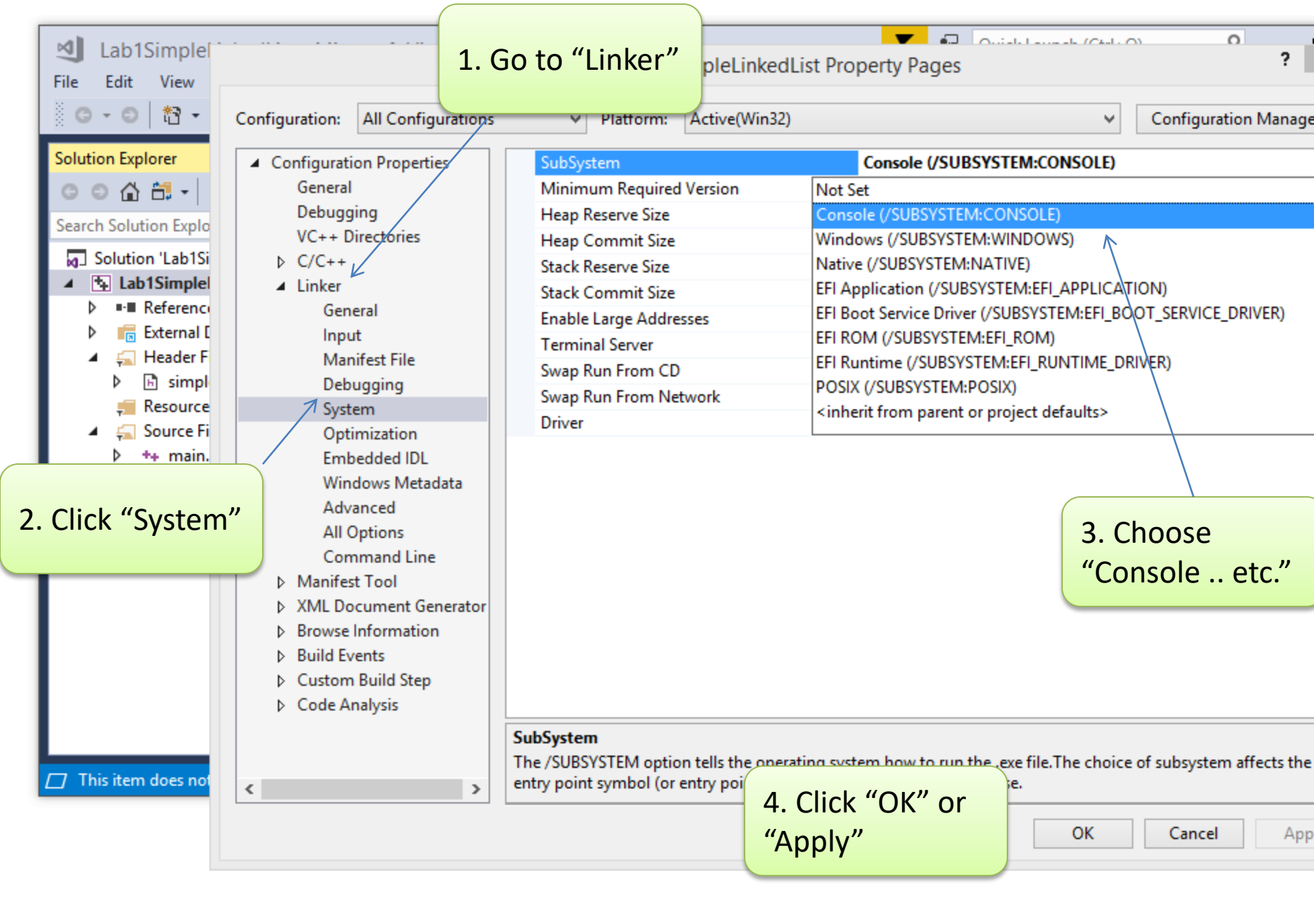
- Something will “flash” in front of your eyes
  - In fact, that is the output (printout) of your code

- Because your program runs, then finished, the console with the output will be closed also
  - (Such a stupid thing)

In Order to View Program Output

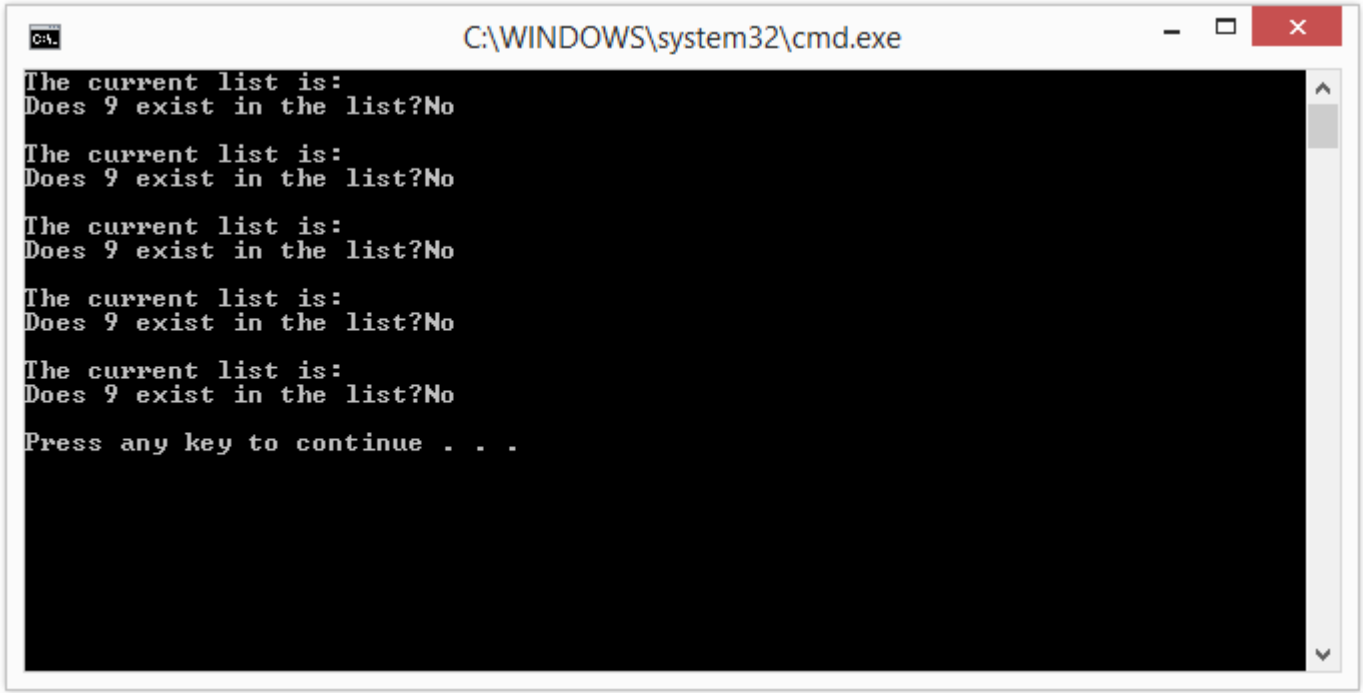
- (You only have to do it once)
- Right Click your project
  - Not the first line but the second
- Select Properties




1. Go to “Linker”
2. Click “System”
3. Choose “Console .. etc.”
4. Click “OK” or “Apply”
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
SubSystem
Console (/SUBSYSTEM:CONSOLE)
Minimum Required Version
Not Set
Heap Reserve Size
Console (/SUBSYSTEM:CONSOLE)
Heap Commit Size
Windows (/SUBSYSTEM:WINDOWS)
Stack Reserve Size
Native (/SUBSYSTEM:NATIVE)
Stack Commit Size
EFI Application (/SUBSYSTEM:EFI_APPLICATION)
Enable Large Addresses
EFI Boot Service Driver (/SUBSYSTEM:EFI_BOOT_SERVICE_DRIVER)
Terminal Server
EFI ROM (/SUBSYSTEM:EFI_ROM)
Swap Run From CD
EFI Runtime (/SUBSYSTEM:EFI_RUNTIME_DRIVER)
Swap Run From Network
POSIX (/SUBSYSTEM:POSIX)
Driver
<inherit from parent or project defaults>
SubSystem
OK
Cancel

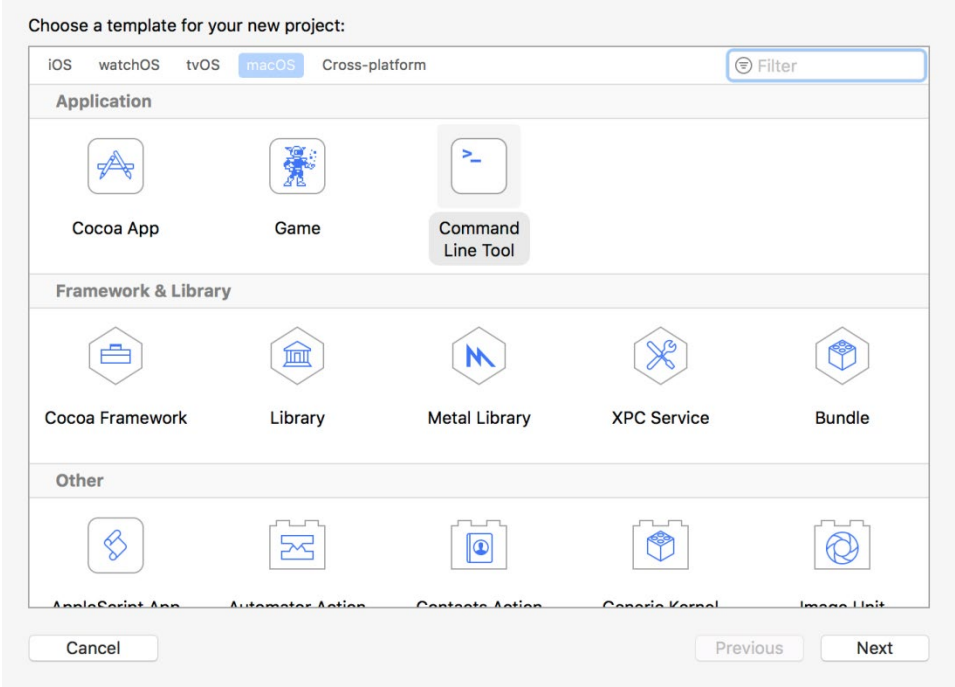
Pause Before Closing Window

- Then there is a line to wait for you to read your output before closing the window



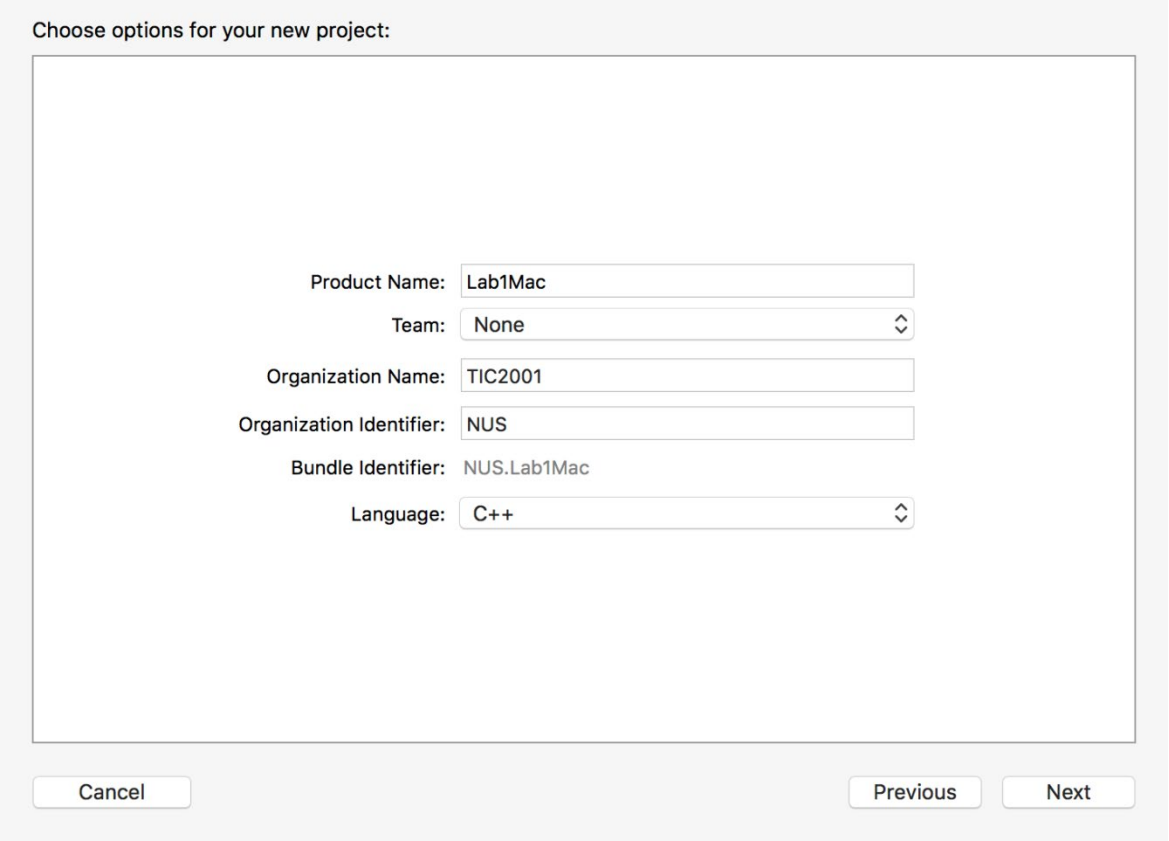
Appendix: Create an Xcode Project

- Start Xcode
- “Create a new Xcode project”
- “Command Line Tool” then “Next”



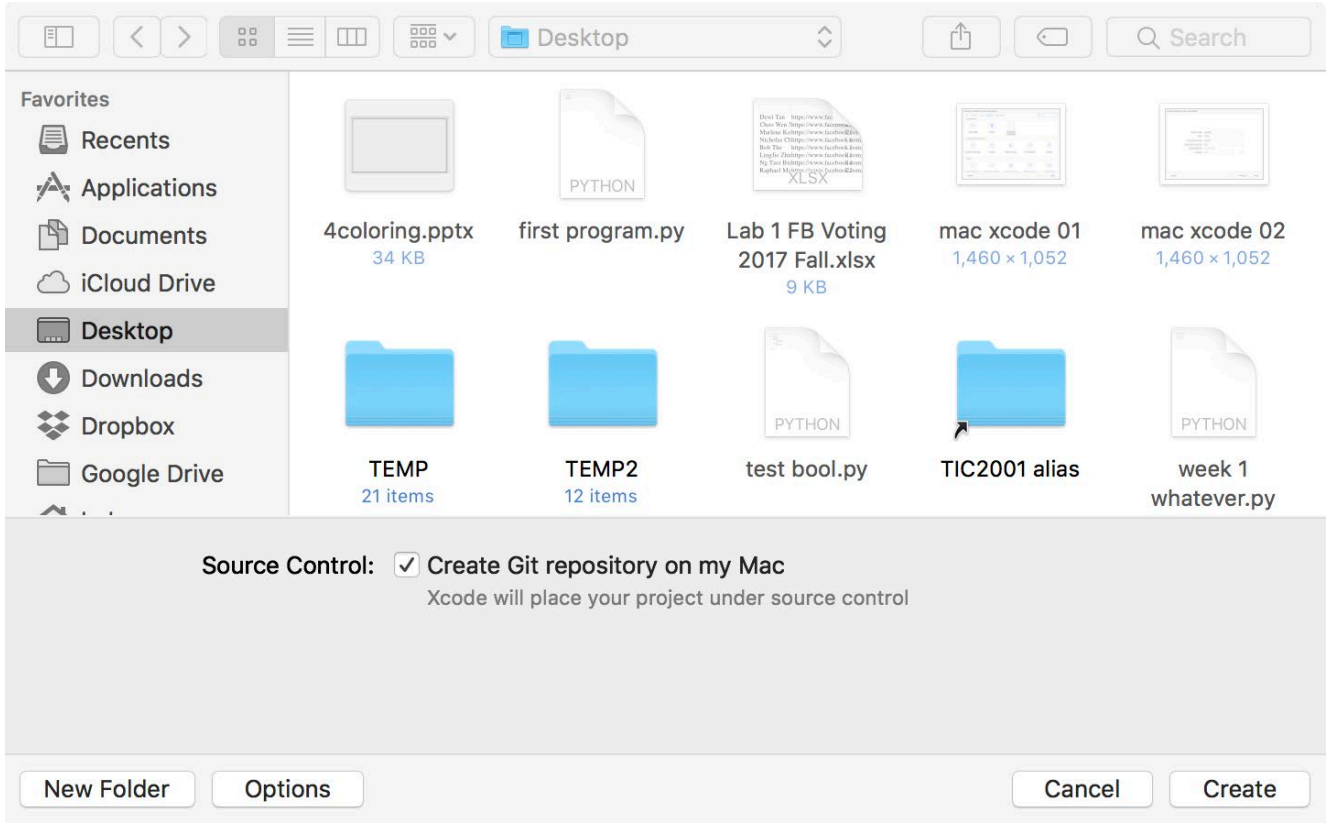
Appendix: Create an Xcode Project

- Name your project name in “Product Name”
- And change other options according to your preference



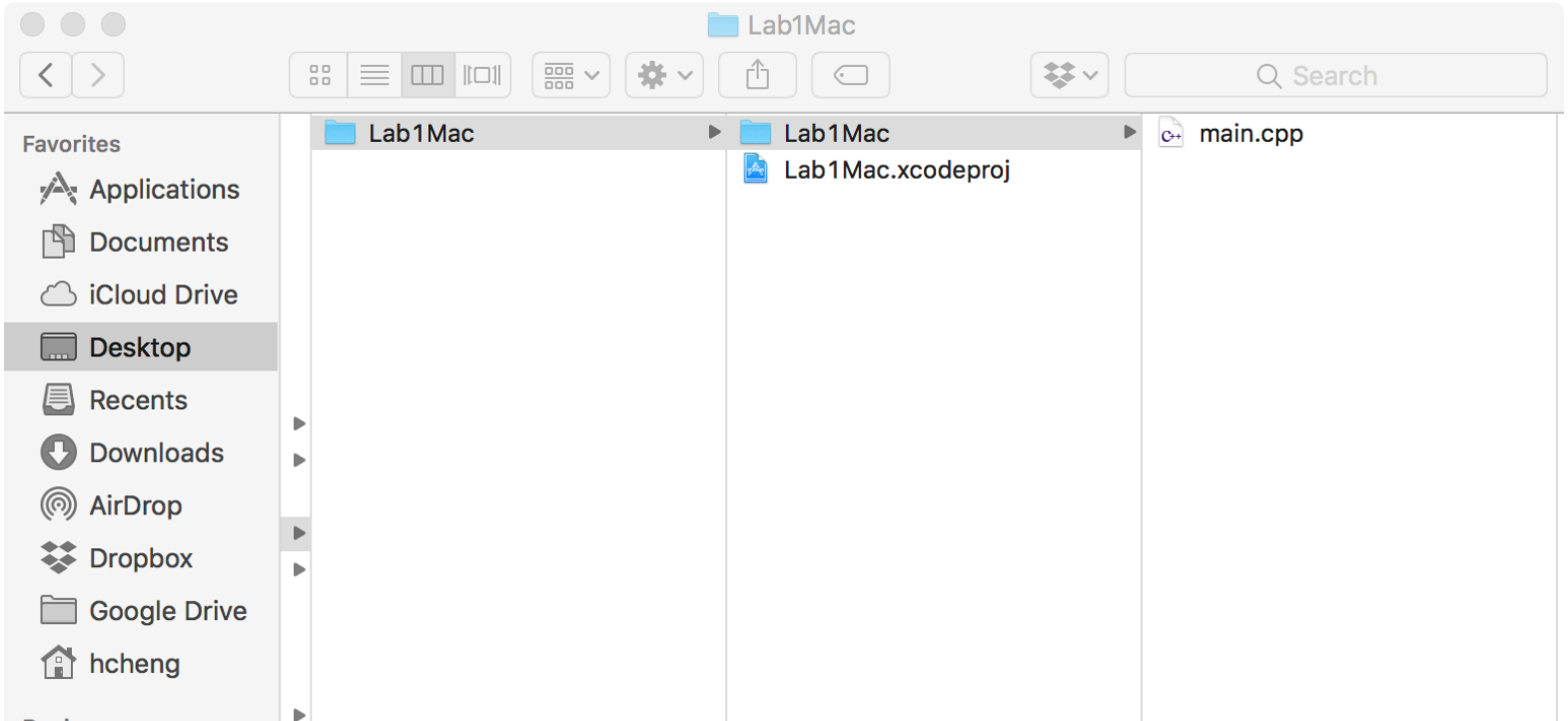
Appendix: Create an Xcode Project

- Then choose a directory/folder to create your project



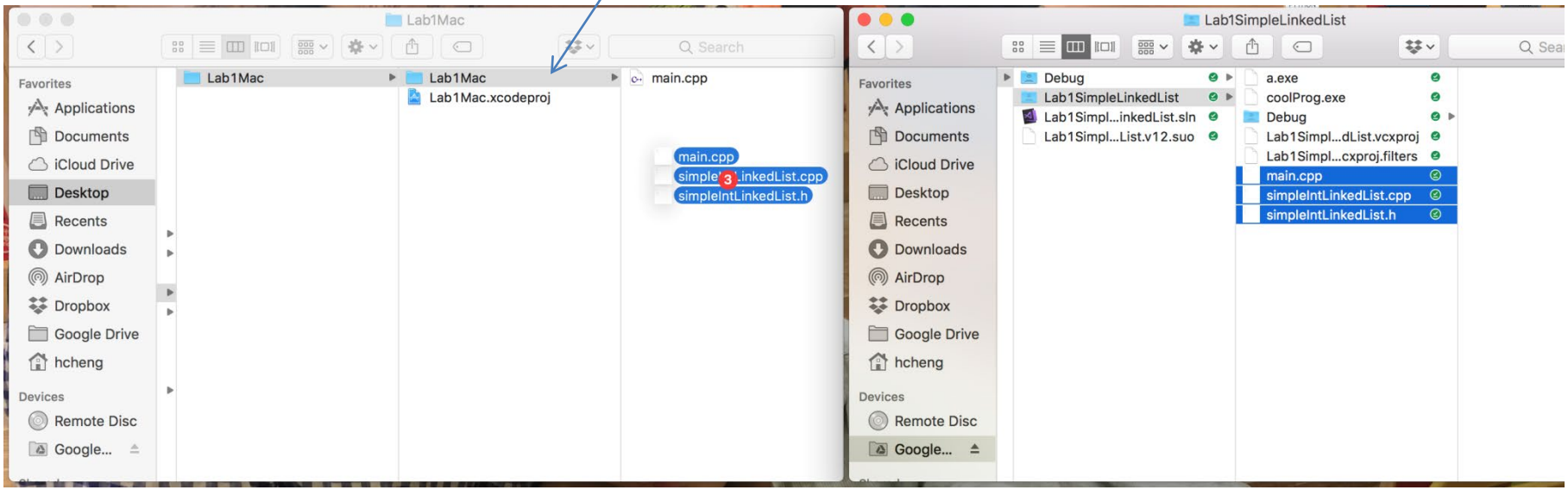
Appendix: Create an Xcode Project

- After doing so, you should have a subfolder with the same name
- And a default main.cpp for you



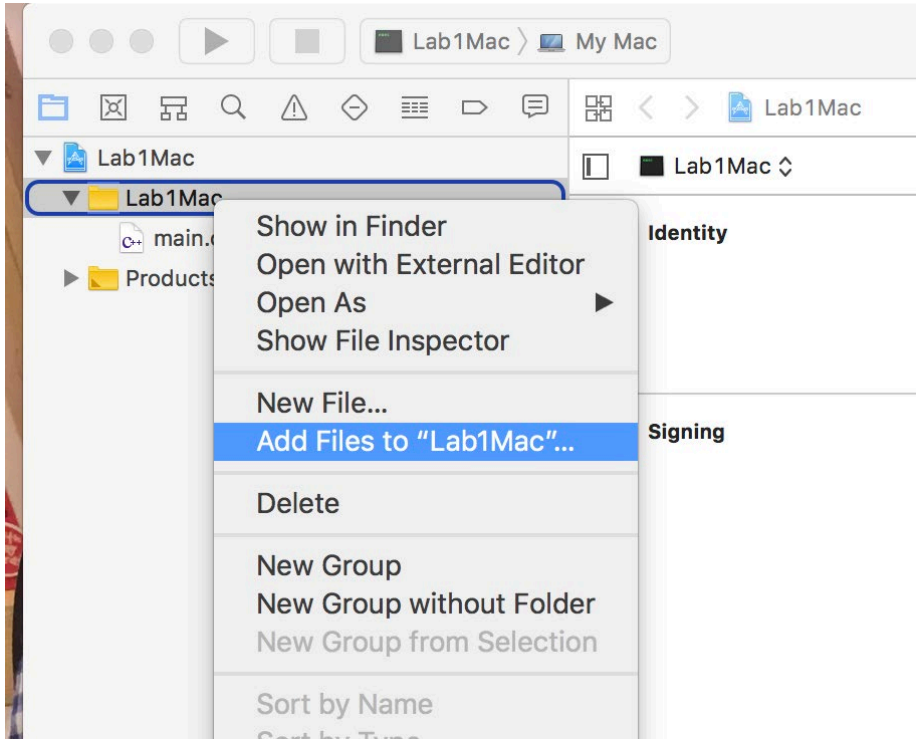
Appendix: Create an Xcode Project

- If you have some existing .h and .cpp files, you can copy them into your project folder
- Usually they should be placed inside the folder besides your .xcodeproj file



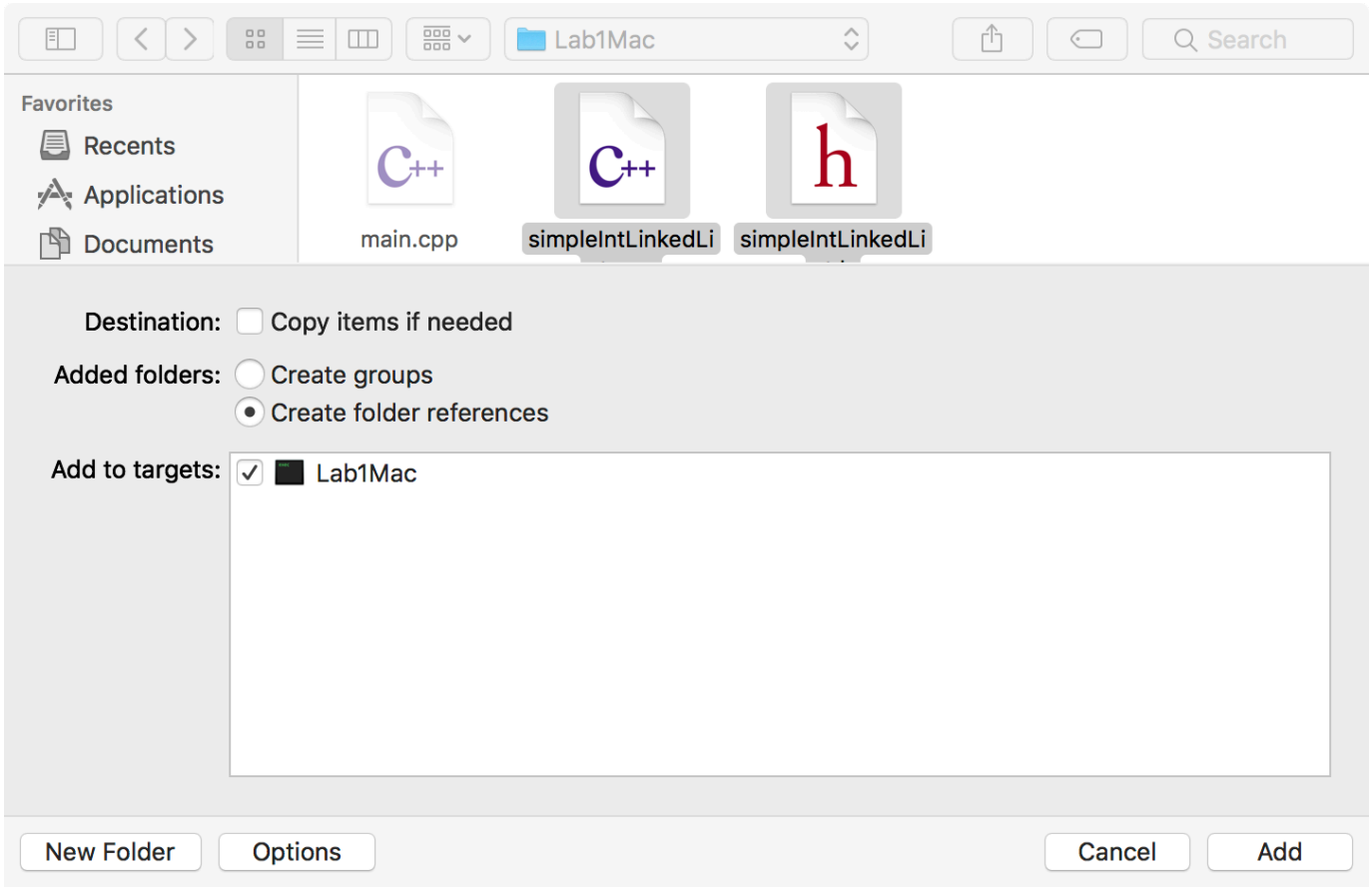
Appendix: Create an Xcode Project

- After opening your .xcodeproj file
- You can add your existing files by clicking the project folder inside Xcode and “Add Files to …”



Appendix: Create an Xcode Project

- Then you can choose what files you want to add into your project



Appendix: Create an Xcode Project

- Finally, you can “Run” your project and the output will be in your output window