

Using Eclipse

Must READ:

Eclipse has far smaller footprint. However, it is more complicated to set up. Just like most OpenSource software, you might have to manually modify some system configuration. If you are not familiar with some simple windows operations such as setting up system environment, or the windows file system, I do not recommend you to install this. You should install Visual Studio instead.

Table of Contents

Foreword	2
File system that you should know	3
Eclipse installation folder	3
Workspace	3
What to download	5
Create a Project	5
To Build your project	7
To enable new console Debugging	8
.gdbinit – config file	8
How to set it up:	8
Build a system containing multiple files	10
A typical User Project path	10
Files structure of a project	10
What if you want to add in your own header file	11
What if you want to include a user library	12
Run your own *.exe which needs user library in a shell/command prompt	12

FOREWORD

Eclipse is an open source IDE running on java. From time to time, the UI may appear a bit strange. Do expect latency sometimes. Also, do not expect all versions being backward compatible. My rule of thumb: the latest usually not the greatest, and can be buggy.

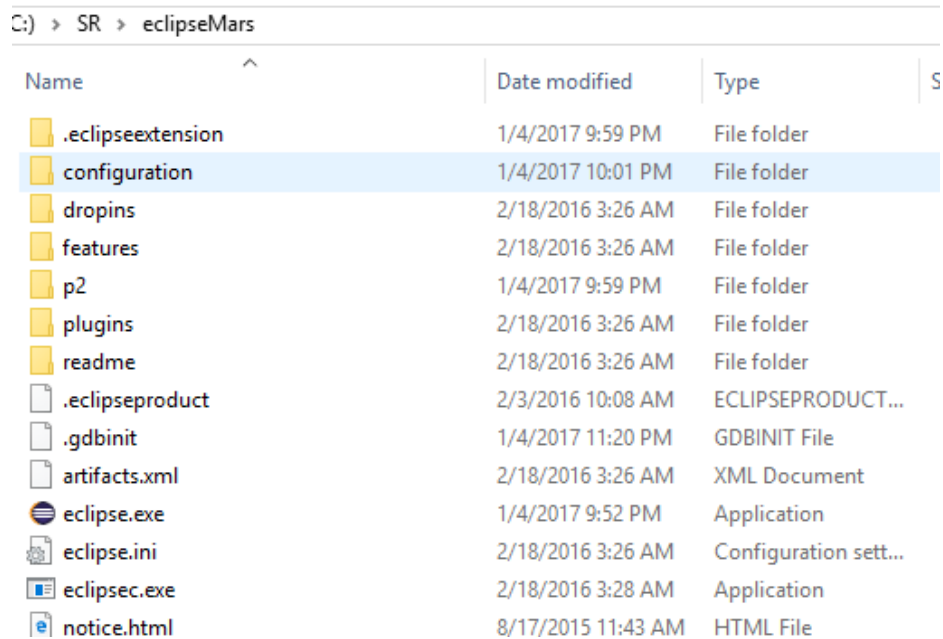
Each version of Eclipse is independent from each other. Thus, it is very easy to migrate in-between without worrying about linking to the wrong libraries; unlike VS does. You just need to make sure you do not go to mess around the default configuration specific to eclipse's installation tree. Not saying that you cannot, but you must know exactly what you are changing and the impact of the change as well.

This document covers the most needed features that you will use in our Algorithms in C/C++ track. To learn more about the IDE, you should [consult its online document](#).

FILE SYSTEM THAT YOU SHOULD KNOW

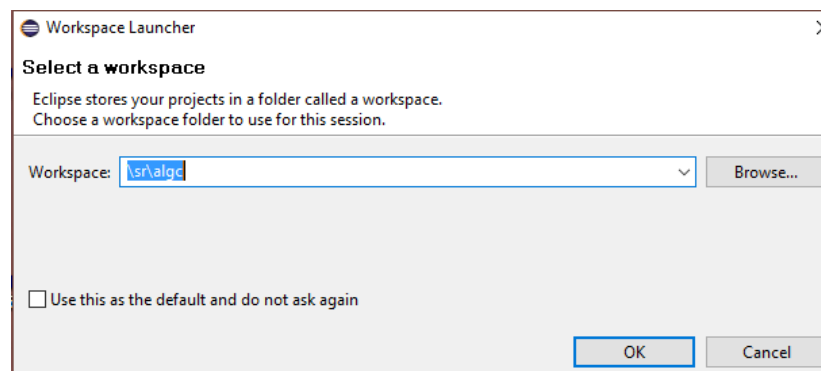
ECLIPSE INSTALLATION FOLDER

In this case, the software is installed under `c:\SR\eclipseMars`



Name	Date modified	Type
.eclipseextension	1/4/2017 9:59 PM	File folder
configuration	1/4/2017 10:01 PM	File folder
dropins	2/18/2016 3:26 AM	File folder
features	2/18/2016 3:26 AM	File folder
p2	1/4/2017 9:59 PM	File folder
plugins	2/18/2016 3:26 AM	File folder
readme	2/18/2016 3:26 AM	File folder
.eclipseproduct	2/3/2016 10:08 AM	ECLIPSEPRODUCT...
.gdbinit	1/4/2017 11:20 PM	GDBINIT File
artifacts.xml	2/18/2016 3:26 AM	XML Document
eclipse.exe	1/4/2017 9:52 PM	Application
eclipse.ini	2/18/2016 3:26 AM	Configuration sett...
eclipsec.exe	2/18/2016 3:28 AM	Application
notice.html	8/17/2015 11:43 AM	HTML File

WORKSPACE



This is basically a folder to save your work including specific configuration (profile) and your files. E.g.

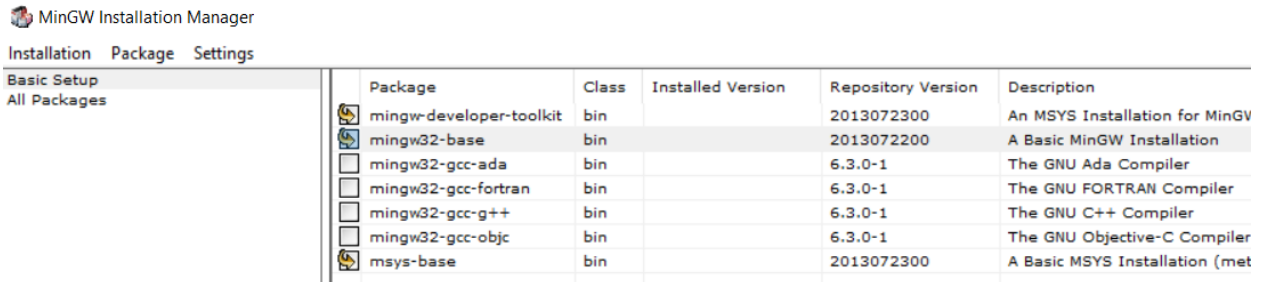
- Project “Chap8prob1 “ – contains a single source file “prob1.cpp”; or
- Project “Chap7” – contains multiple files chap7main.cpp, ch7_func1.cpp, ch7_func2.cpp, myHeader.h, etc.

Note: the profile set under this “workspace – “\sr\algc” (in the example) will apply to all projects created under this workspace. Go to file explorer to check out the projects list to a get a better understanding regarding the workshop layout.

WHAT TO DOWNLOAD

1. The latest Java JDK which we know it works with Eclipse was JDK 1.8.x
2. Download Eclipse. The last one we worked on was Eclipse oxygen.
3. Download the compiler MinGW Installation manager:

<https://sourceforge.net/projects/mingw/files/latest/download?source=files>

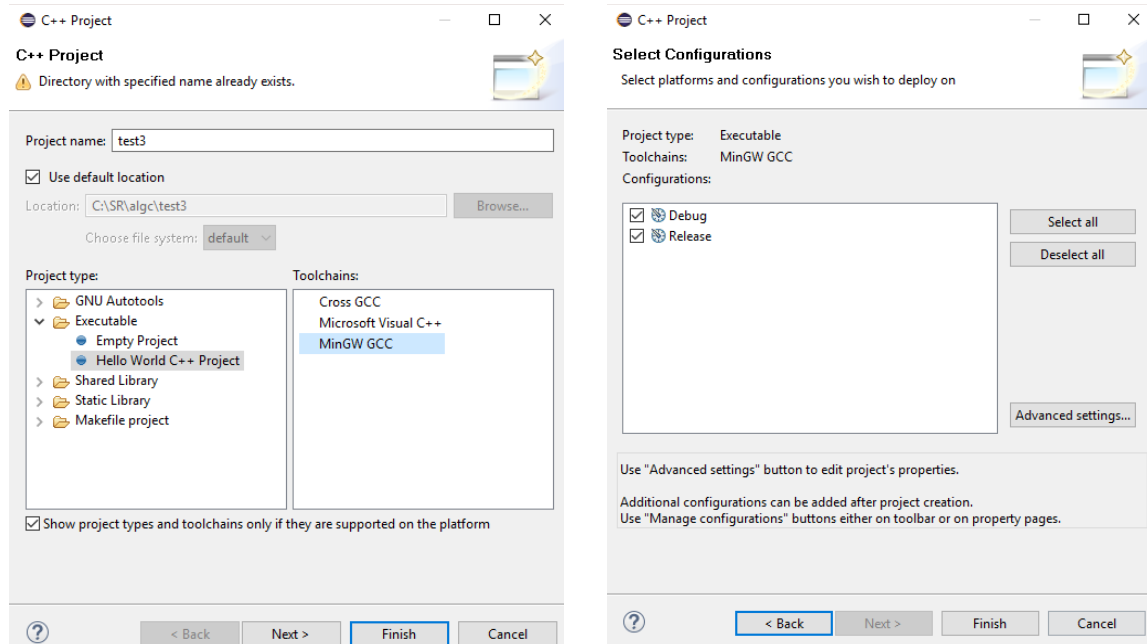


Then : Installation -> Apply Changes

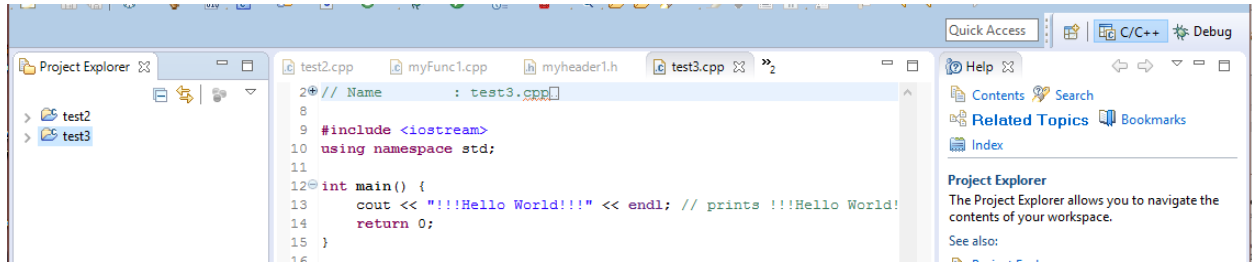
CREATE A PROJECT

: **FILE > NEW > C++ PROJECT**

: You may create *.cpp or *.h files



Then, you will see something like the following:

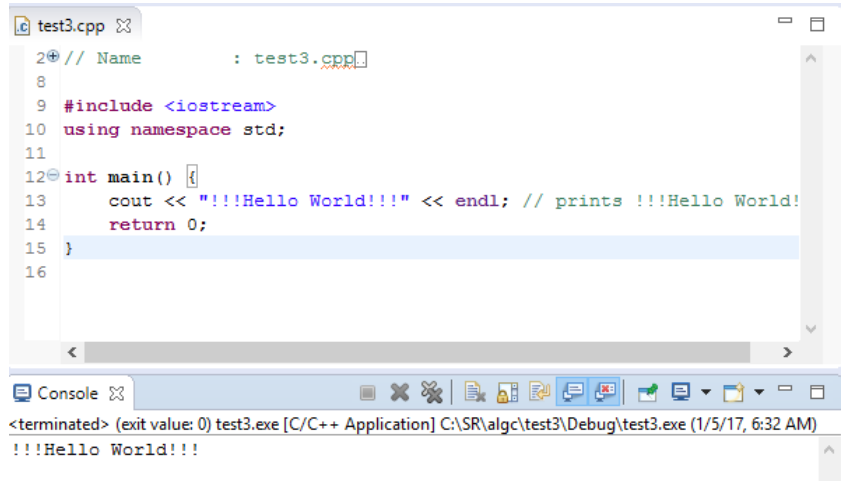


TO BUILD YOUR PROJECT

Click: **PROJECT > BUILD PROJECT**

To execute: **CTRL + F11** or **CLICK RUN > RUN**  or **RUN > DEBUG** 

Watch the console display below:




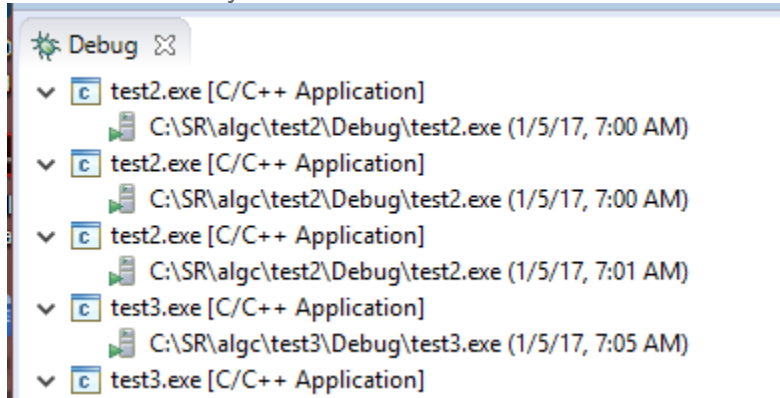
Possible issue:

- 1) If the console window does not display, do reset “Perspective“:

WINDOW > PERSPECTIVE > RESET PERSPECTIVE

- 2) Got “Permission denied to build the *.exe.
e.g. cannot open output file test3.exe: Permission denied test3

- Toggle to Debugger window -  |  **Debug**
- You will a list of your *.exe on the left :



- They are copies of the executable currently active. You need to terminate them
- You may terminate one instance to all instances. Right click on it, you can, eg. Do:

TERMINATE/DISCONNECT ALL

(this is definitely by far better than the Visual Studio Hanging Executable; in that case, there is no easy way but to restart the VS before you can rebuild your program.)

TO ENABLE NEW CONSOLE DEBUGGING

.GDBINIT – CONFIG FILE

This file should exist in your eclipse installation. If this file does not exist, you should create one using a simple text notepad.

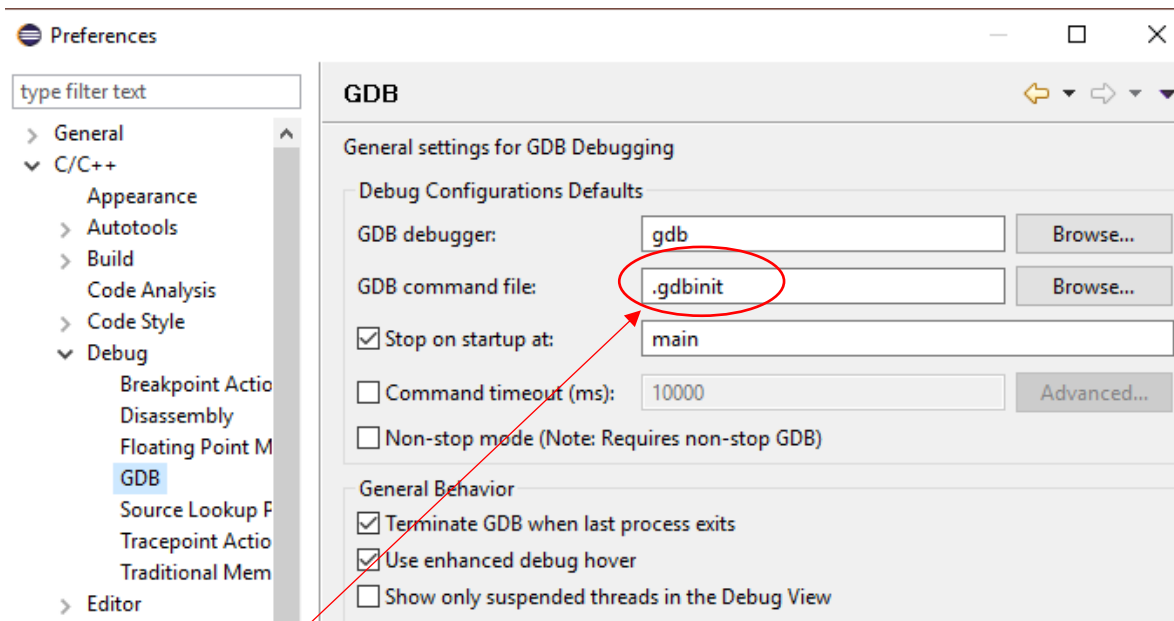
Type in : `set new-console on`

Important: the file name must be named exactly as described, prefixed with “.”. All lower case. No file extension.

This file must exist in order for you to do a new console debugger method. This often helps in terms of detailed debugging with users standard input.

HOW TO SET IT UP:

: **WINDOW > C++ > DEBUG > GDB**

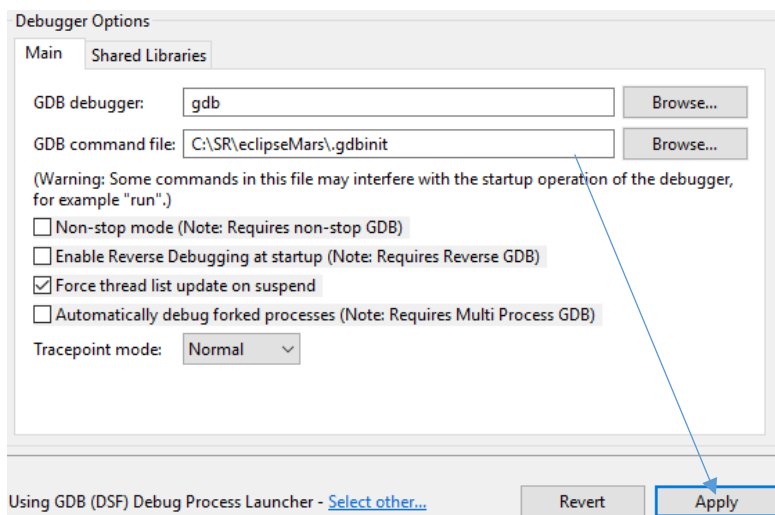
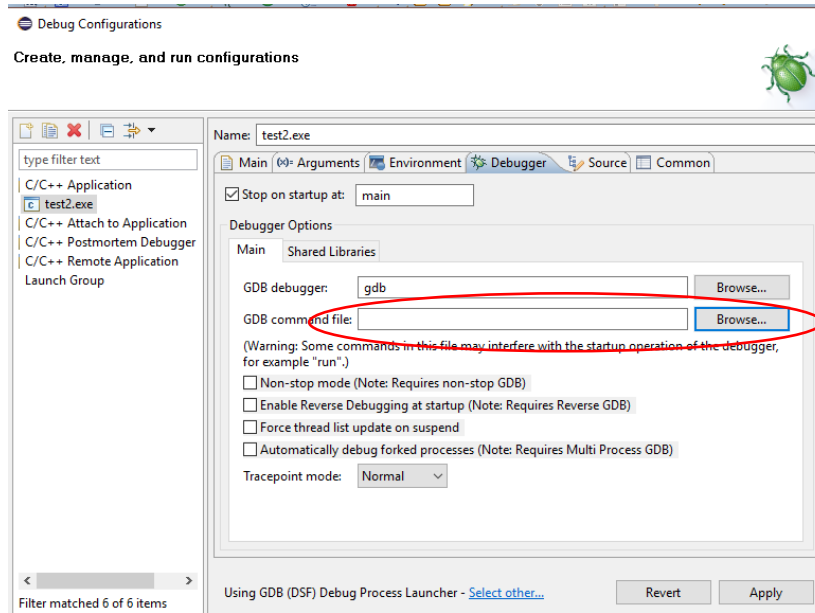


Change that to .gdbinit file located in the eclipse installation folder.

Do note: what you change here will be saved as part of “profile belong to the workspace folder” which you just entered as shown earlier.

Another note:

If you set this .gdbinit under **RUN > DEBUG CONFIGURATION** (see below), the new console debugging profile will only be applicable to the current project, not all projects in the workshop.

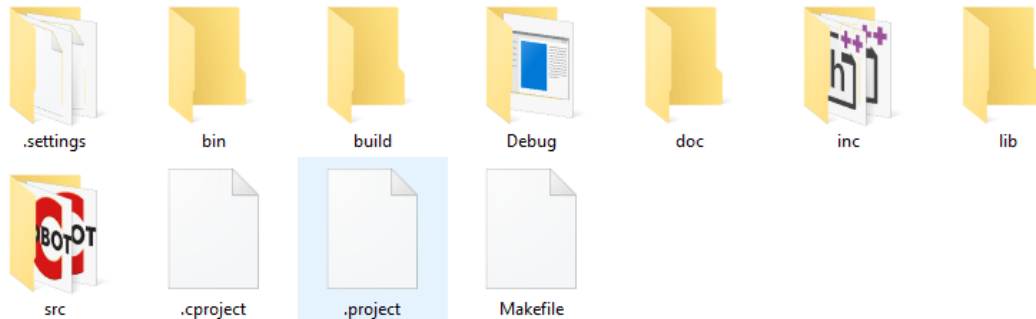


Possible issue:

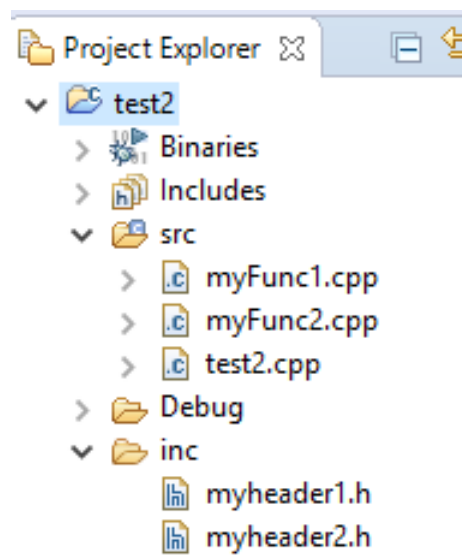
Sometimes, the Apply button is disabled. You will just have to delete the GDB command file entry and add it back in. The Apply button will be enabled again.

BUILD A SYSTEM CONTAINING MULTIPLE FILES

A TYPICAL USER PROJECT PATH



FILES STRUCTURE OF A PROJECT



```
// sample main file – test2.cpp
#include <stdio.h>

// < ... > only works if you have added the path into the "include
//      symbol path as shown above
#include <myheader2.h>

// or direct reference
#include "../include/myheader1.h"

int main() {
    printf("%d! = %d\n", MaxNum1, factorial1(MaxNum1));
    printf("%d! = %d\n", MaxNum2, factorial2(MaxNum2));
    return 0;
}
```

```
/* myheader1.h sample

#ifndef MYHEADER1_H_
#define MYHEADER1_H_

extern const int MaxNum1;
int factorial1(int);

#endif /* MYHEADER1_H_ */
```

```
/* myheader2.h sample

#ifndef MYHEADER1_H_
#define MYHEADER1_H_

extern const int MaxNum2;
int factorial2(int);

#endif /* MYHEADER2_H_ */
```

```

/* myFunc1.cpp Sample */

#include "../include/myheader1.h"

const int MaxNum1 = 11;

int factorial1(int n)
{
    if (n==2)      return 2;
    return n * factorial1(n-1);
}
    
```

```

/* myFunc2.cpp Sample */

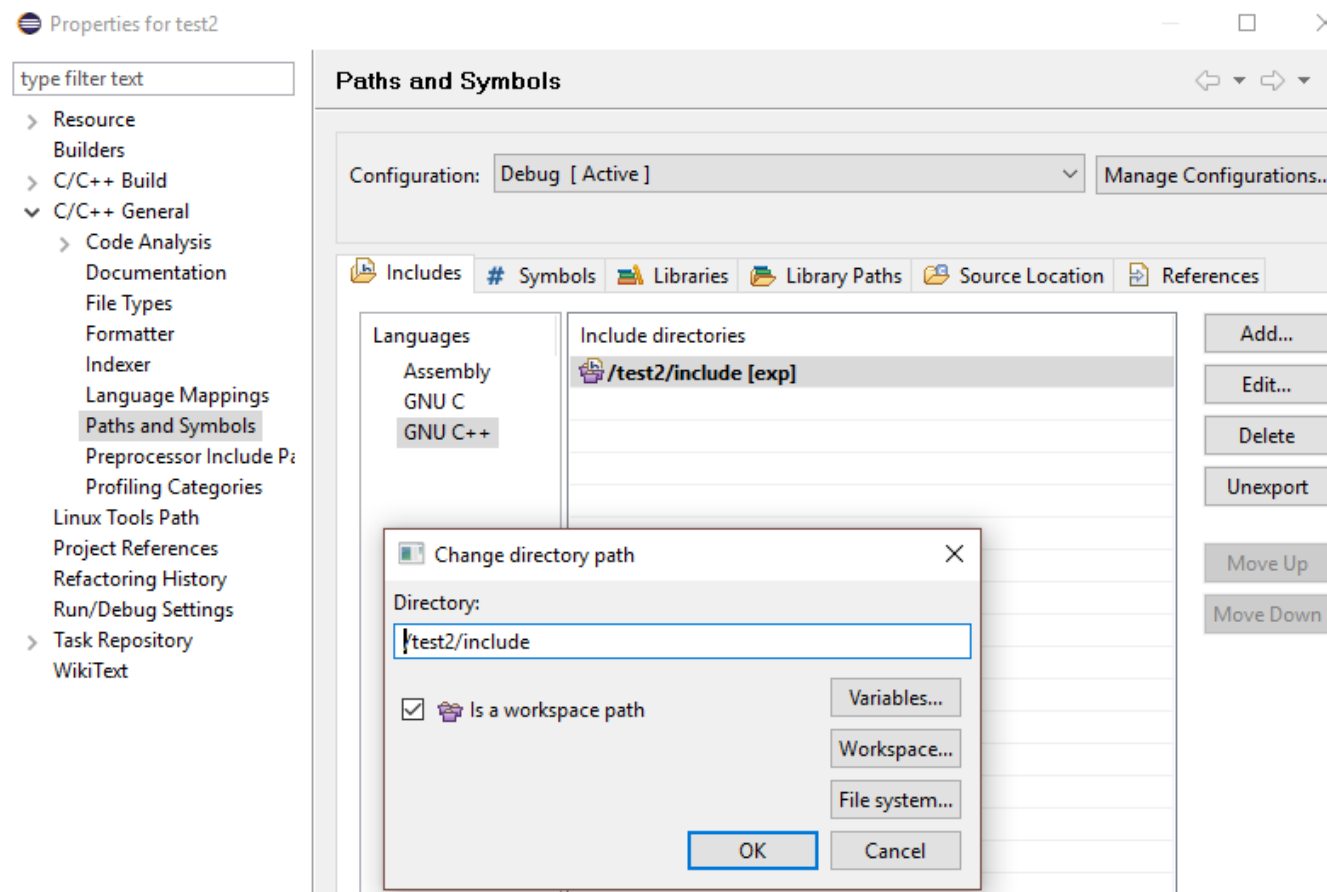
#include "../include/myheader2.h"

int MaxNum2 = 12;

int factorial2(int n)
{
    int result;
    for (result=n; n>2; n--)
        result = result * (n-1);
    return result;
}
    
```

WHAT IF YOU WANT TO ADD IN YOUR OWN HEADER FILE

You will need to add include path.



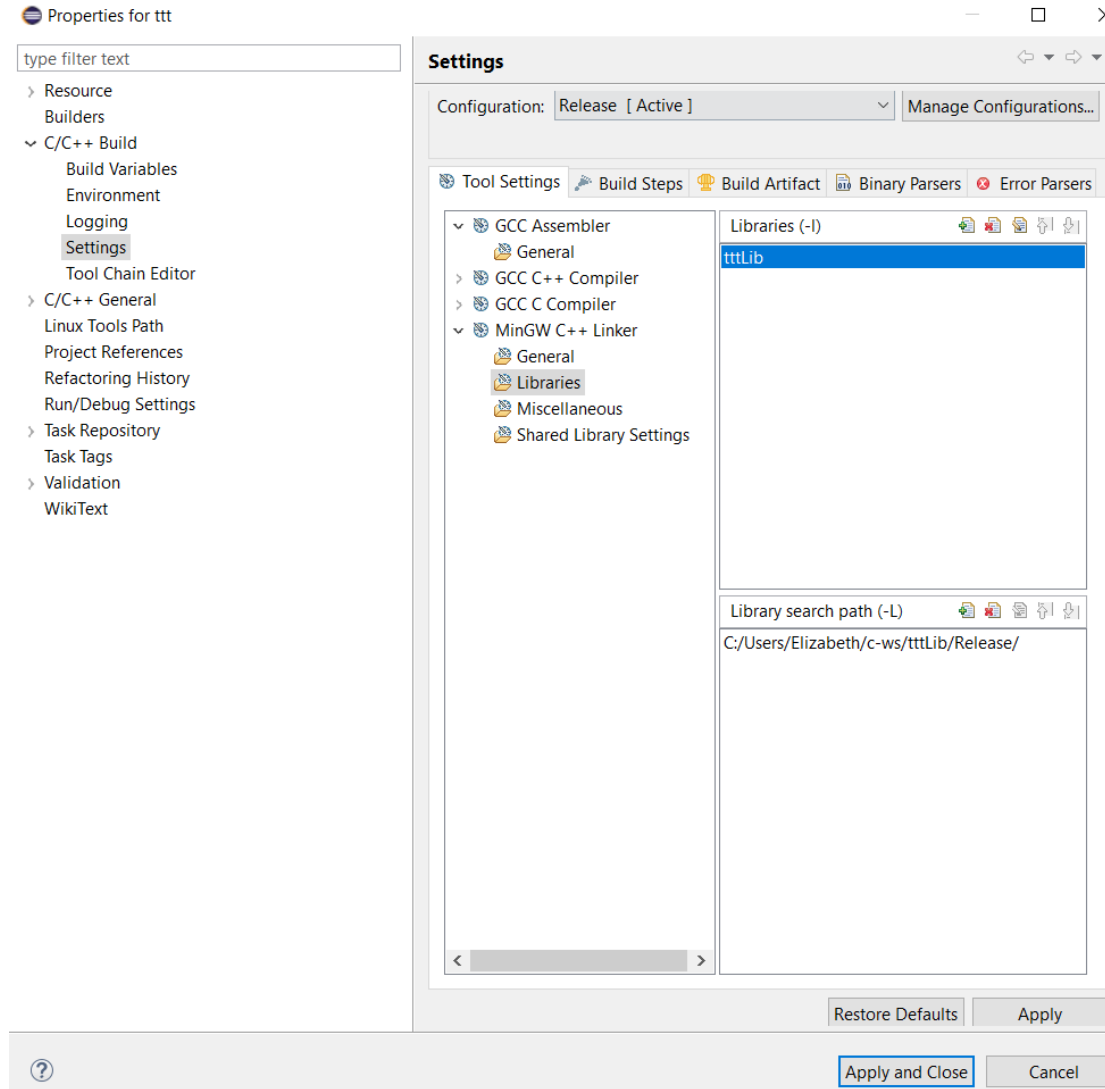
Make sure you will always click **Apply** before you close this window.

You may select it to apply to all configuration, or just to the current project.

If you want to apply to the all projects, you will need to do “Export” , “Refresh”., and “Rebuild” after you add in the include path(s).

WHAT IF YOU WANT TO INCLUDE A USER LIBRARY

- Then, Add library path and user library label.
- Assuming the user library is called libttdLib.dll, and is stored in the folder “C:\Users\Elizabeth\c-ws\ttdLib\Release” :



Make sure you will always click **Apply** before you close this window.

RUN YOUR OWN *.EXE WHICH NEEDS USER LIBRARY IN A SHELL/COMMAND PROMPT

- Assuming you put libttdLib.dll in path “C:\Users\you\c-ws\ttdLib\Release”
- set path=%path%;C:\Users\you\c-ws\ttdLib\Release;

After that, you can run your *.exe.