CS201: Recitation 1

GNU Compiler, Server, Makefile

(not writing any code)

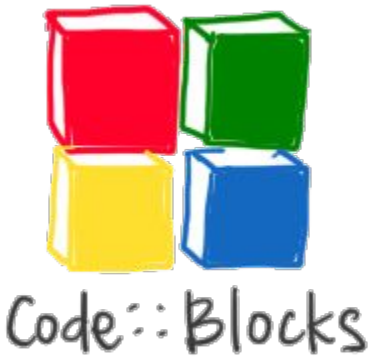
Content

1) Downloading an IDE with GNU compiler
2) Running your first simple C++ code
3) Running your first simple C++ on the server
4) Headers and running projects with multiple files
5) Using Valgrind to check for memory leaks
6) Makefile, making compiling your project a bit easier
7) Debugging your code in IDE

1. Download an IDE

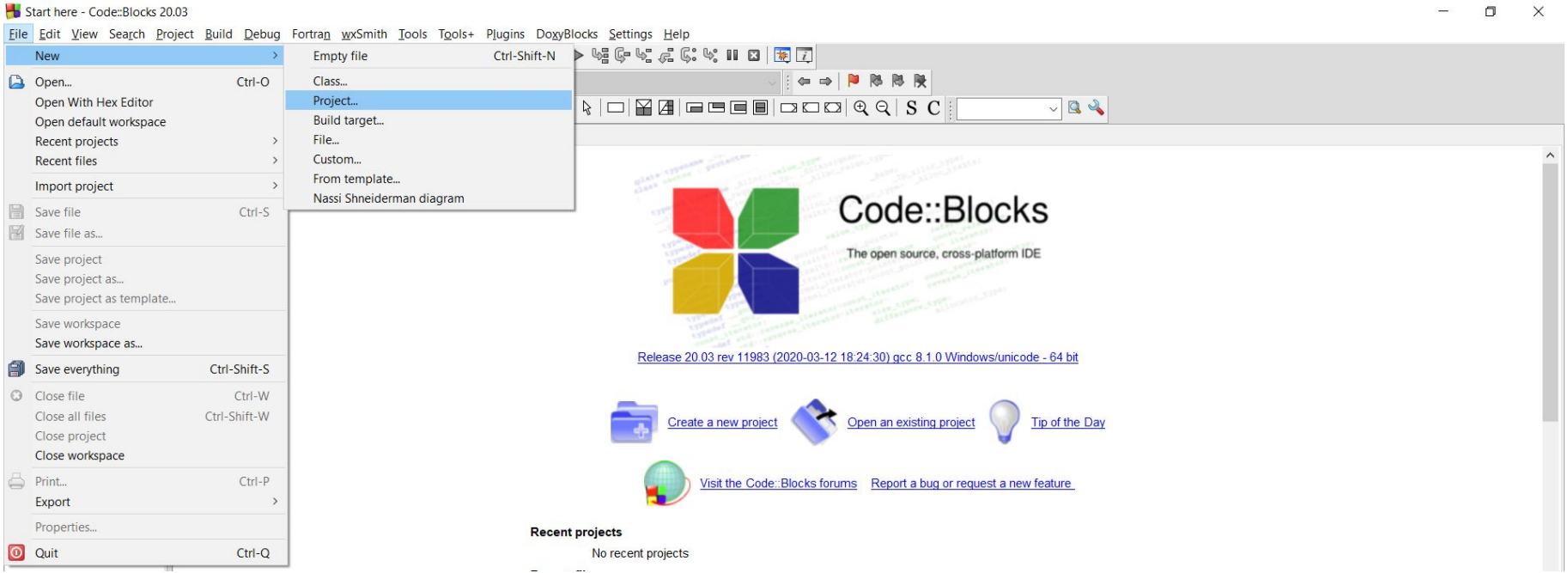
1) One good option is CodeBlocks:
   https://www.codeblocks.org/downloads/binaries/

2) For windows make sure you download ‘mingw’ option



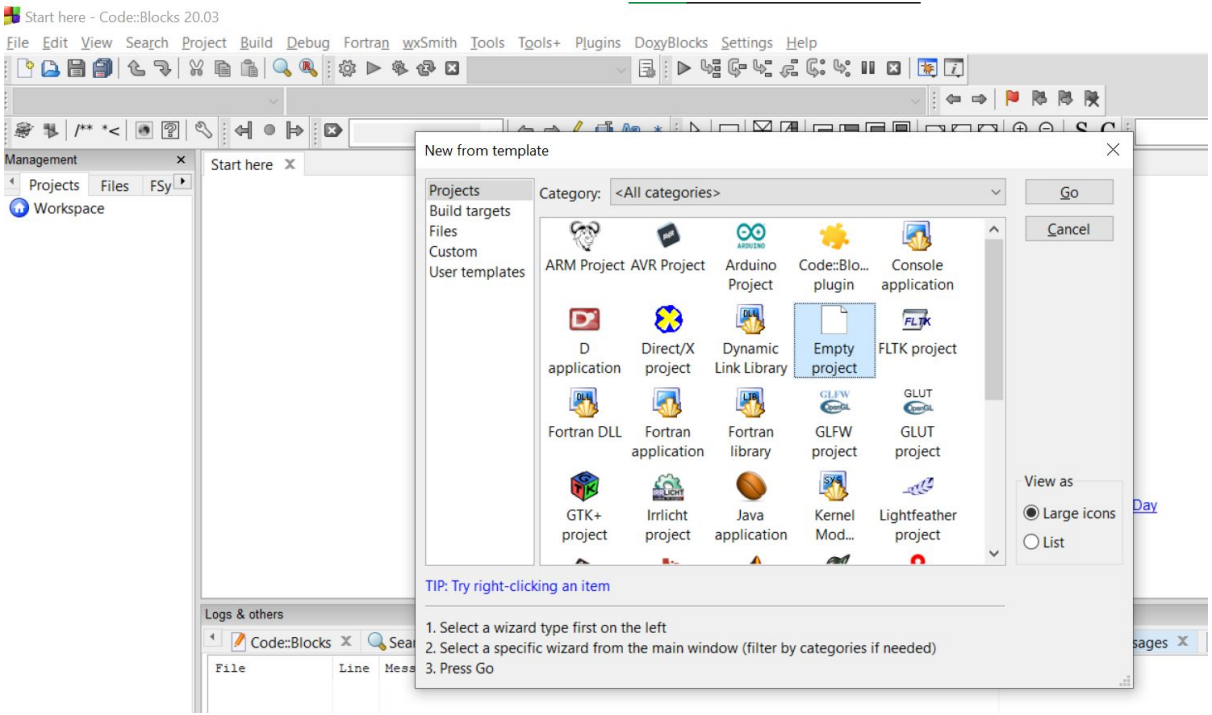
2. Create and run a simple C++ code as a part of a project

In upper left corner click: File > New > Project



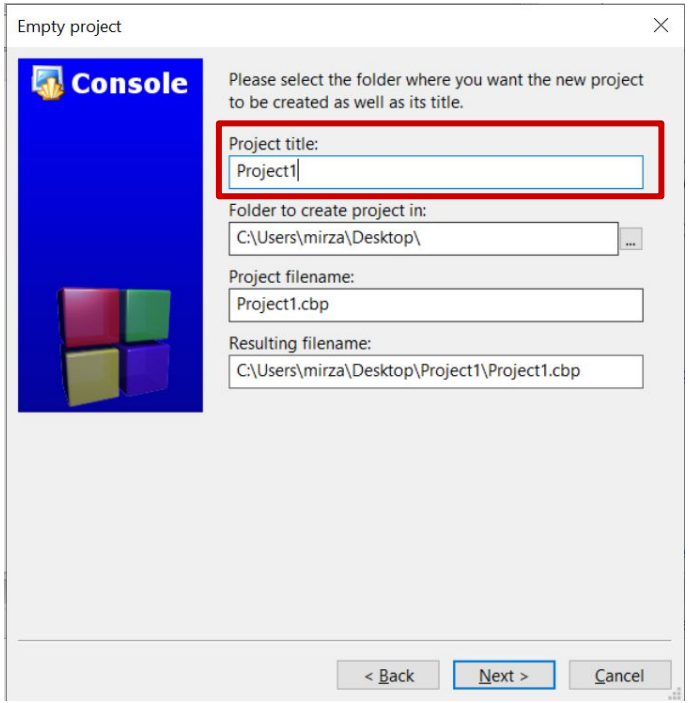
2. Create and run a simple C++ code as a part of a project

Choose Empty Project

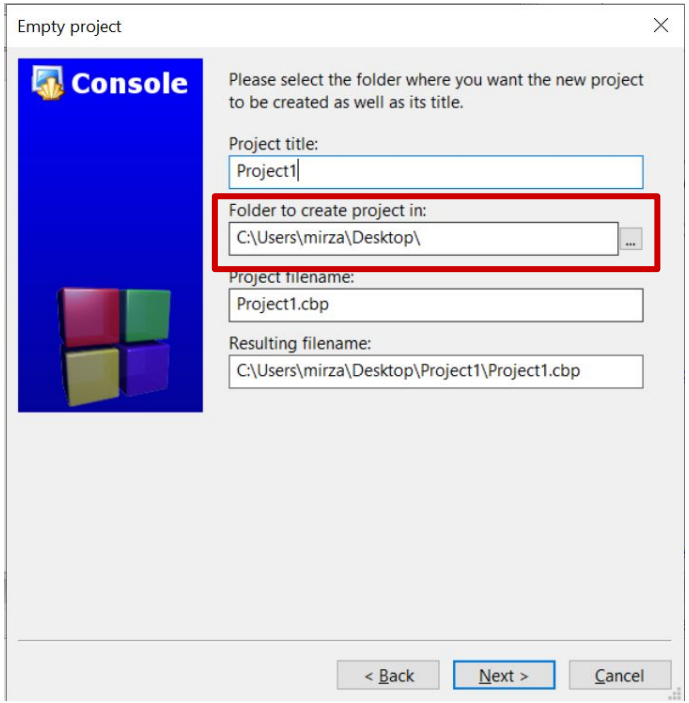


2. Create and run a simple C++ code as a part of a project

Give your project a name

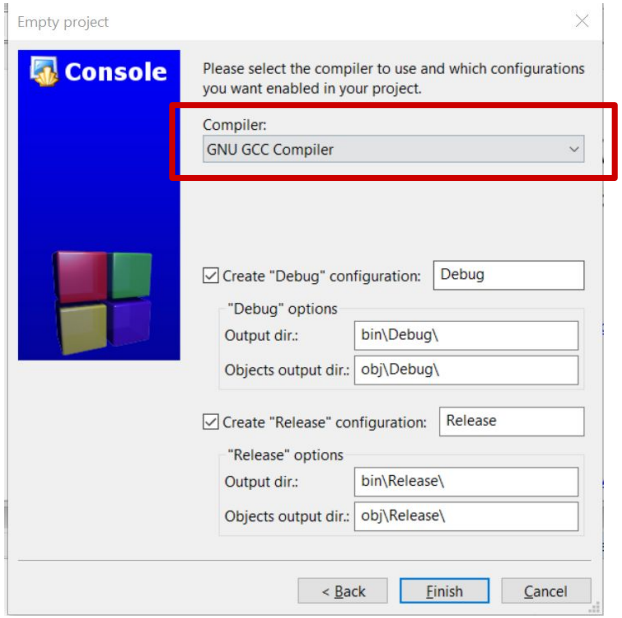


And choose a project folder



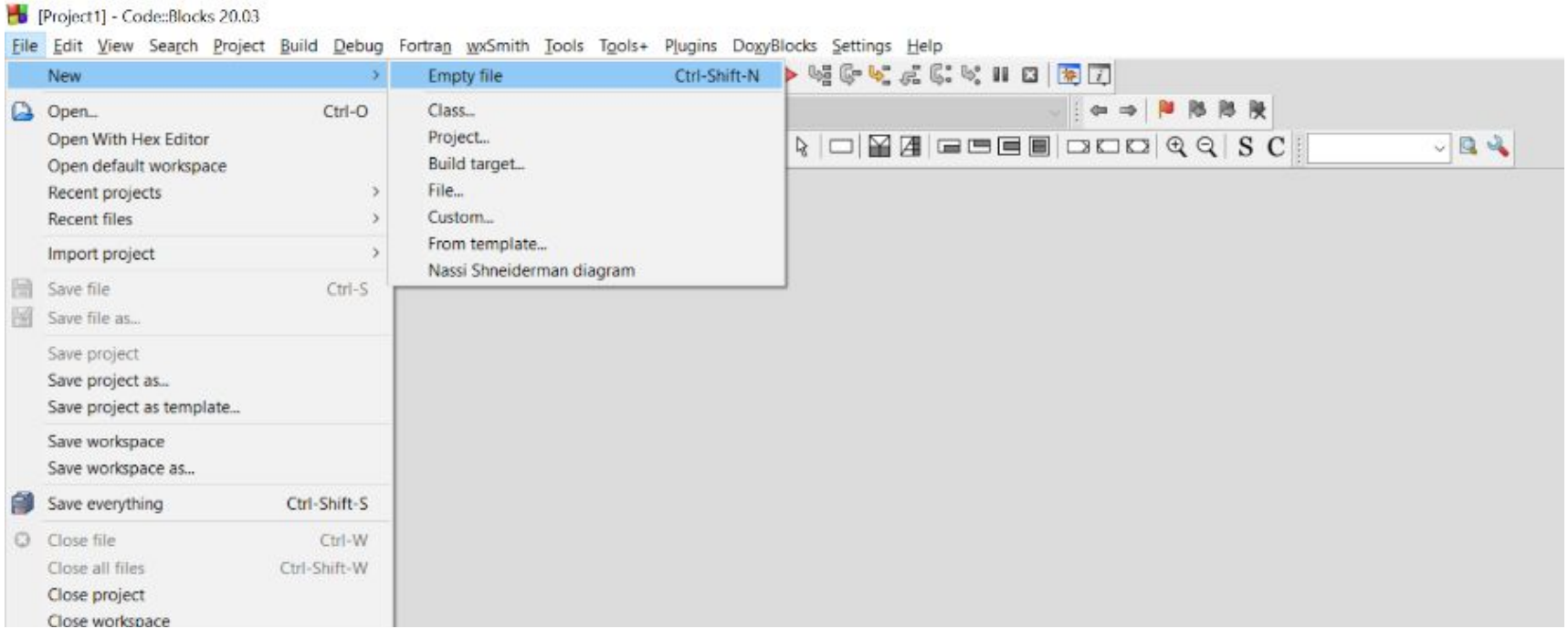
2. Create and run a simple C++ code as a part of a project

VERY IMPORTANT: For this and all your future CodeBlocks projects in CS201 course please make absolutely sure that you choose GNU GCC Compiler. Press Finish.



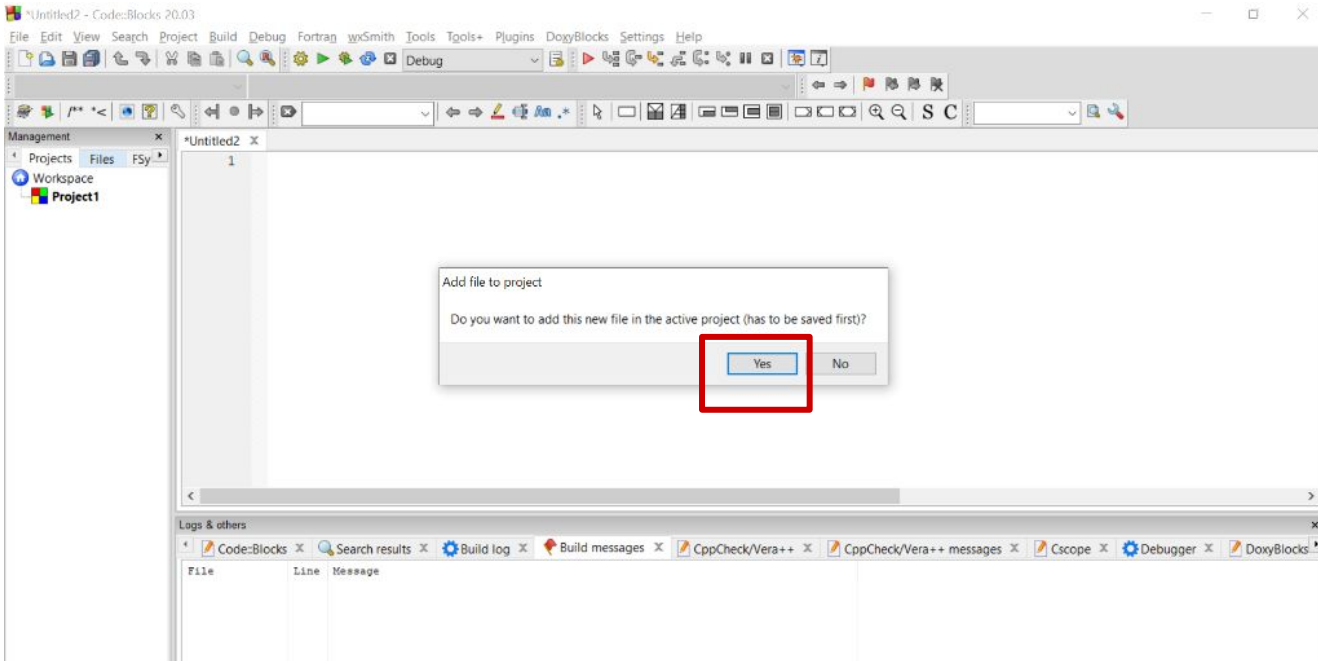
2. Create and run a simple C++ code as a part of a project

Add a C++ file your project: File>New>Empty file



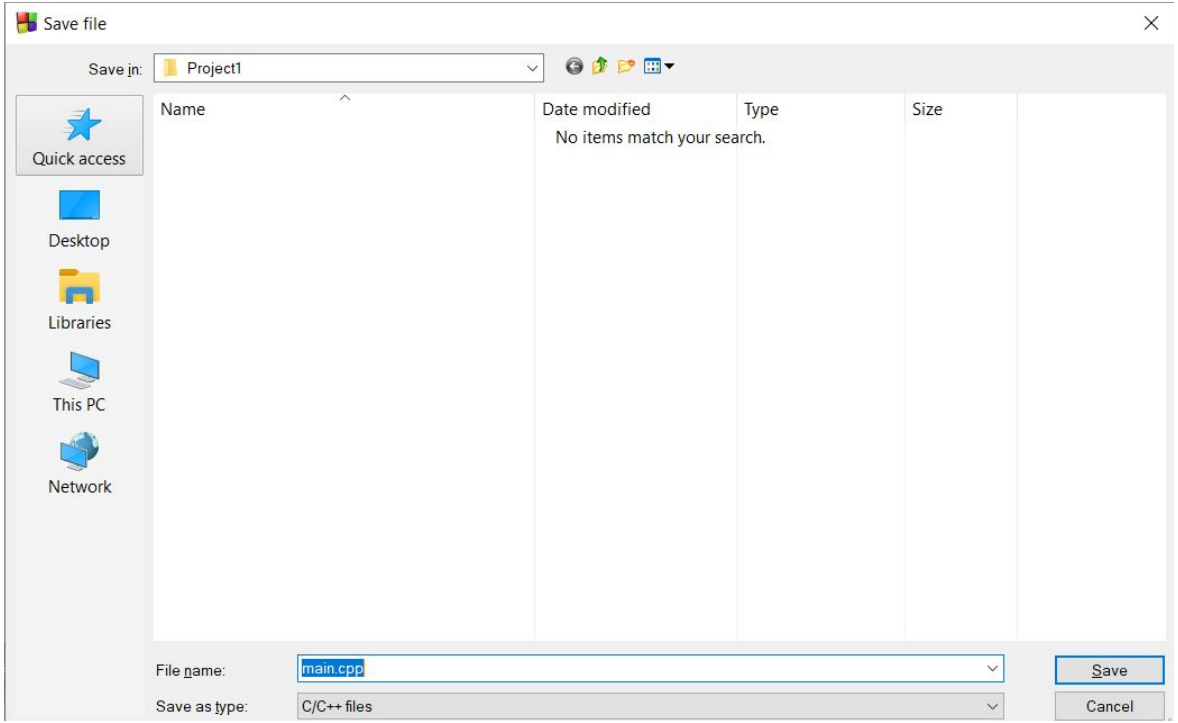
2. Create and run a simple C++ code as a part of a project

Press yes



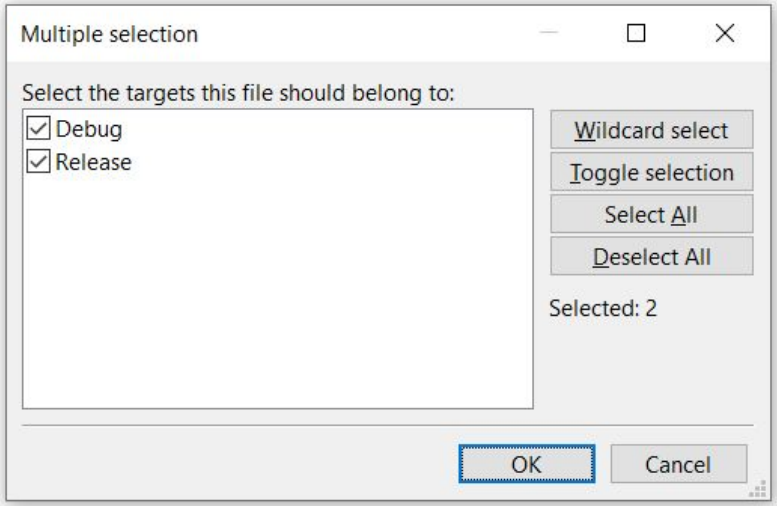
2. Create and run a simple C++ code as a part of a project

Name it ‘main.cpp’ (.cpp is standard c++ file extension)



2. Create and run a simple C++ code as a part of a project

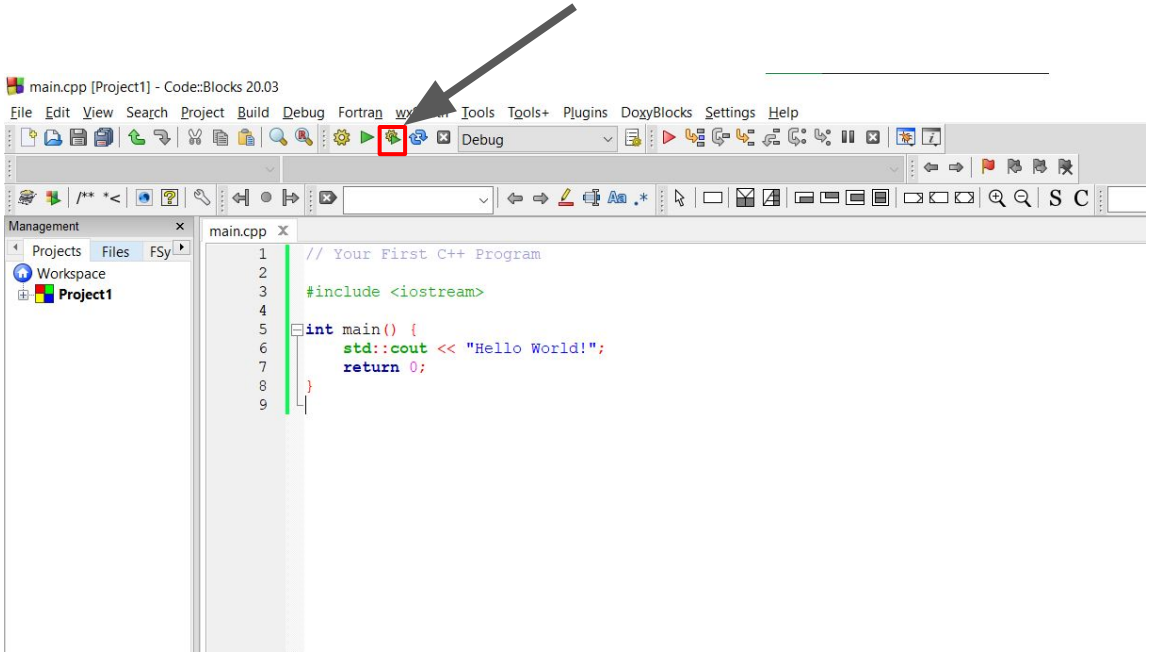
Make sure debug and release options are checked and press ok



2. Create and run a simple C++ code as a part of a project

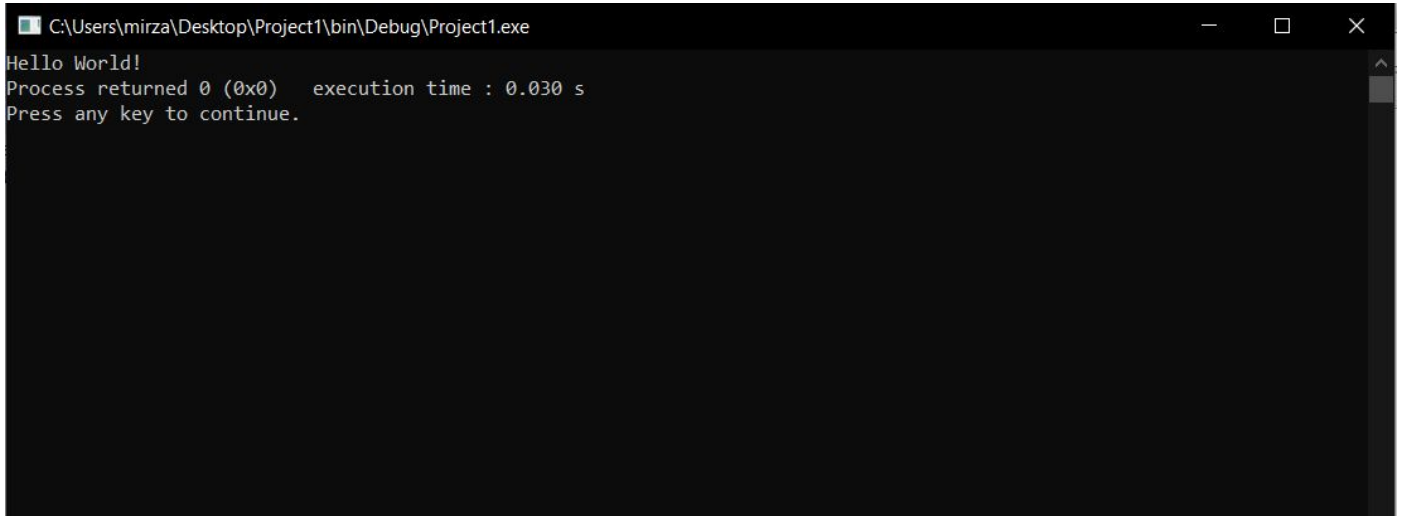
Copy and paste this code and then press 'build and run' icon

```
// Your First C++ Program

#include <iostream>

int main() {
   std::cout << "Hello World!";
   return 0;
}
```



2. Create and run a simple C++ code as a part of a project

If you did everything right, a window will pop up and display something like this.



Congratulations, you ran your first C++ code!

3. Running your code on a remote server

- Unlike your previous coding classes, assignments in CS201 are evaluated not locally but on a remote server
- This means that during grading we will upload your code to a remote server and run it there, and give you a grade based on the output we get
- Therefore, to make sure that there are no unexpected errors and problems when your code is ran on the server, you too have to connect to the server and run your code there before you submit it
- If after submission we cannot run your code on the server, you are likely to lose a good portion of your grade even if your code works perfectly on your computer.

3. Running your code on a remote server

- Luckily, uploading your code to the server and running it there is very simple

- It is a two-step process 1) Uploading your code and 2) Running your code. For each of these processes we use a separate software.

Software for uploading the code: FileZilla
https://filezilla-project.org/download.php?type=client

Software for running the code: Your Command Prompt

Download FileZilla and let's move to the next step. Command prompt is already available on your pc.

3. Running your code on a remote server

- The server we are going to use is located at Bilkent and is called Dijkstra. Think of it as a computer somewhere in one of our buildings which is connected to the bilkent network.

- To connect to this server, you too, must be connected to the Bilkent network. Which means you either need to be using campus internet connection, or connect to your Bilkent VPN if you are not inside the campus.

- Use this link if you don't already have a Bilkent VPN http://web3.bilkent.edu.tr/vpn/

- Again, you don't need VPN if you are connected to Bilkent network.

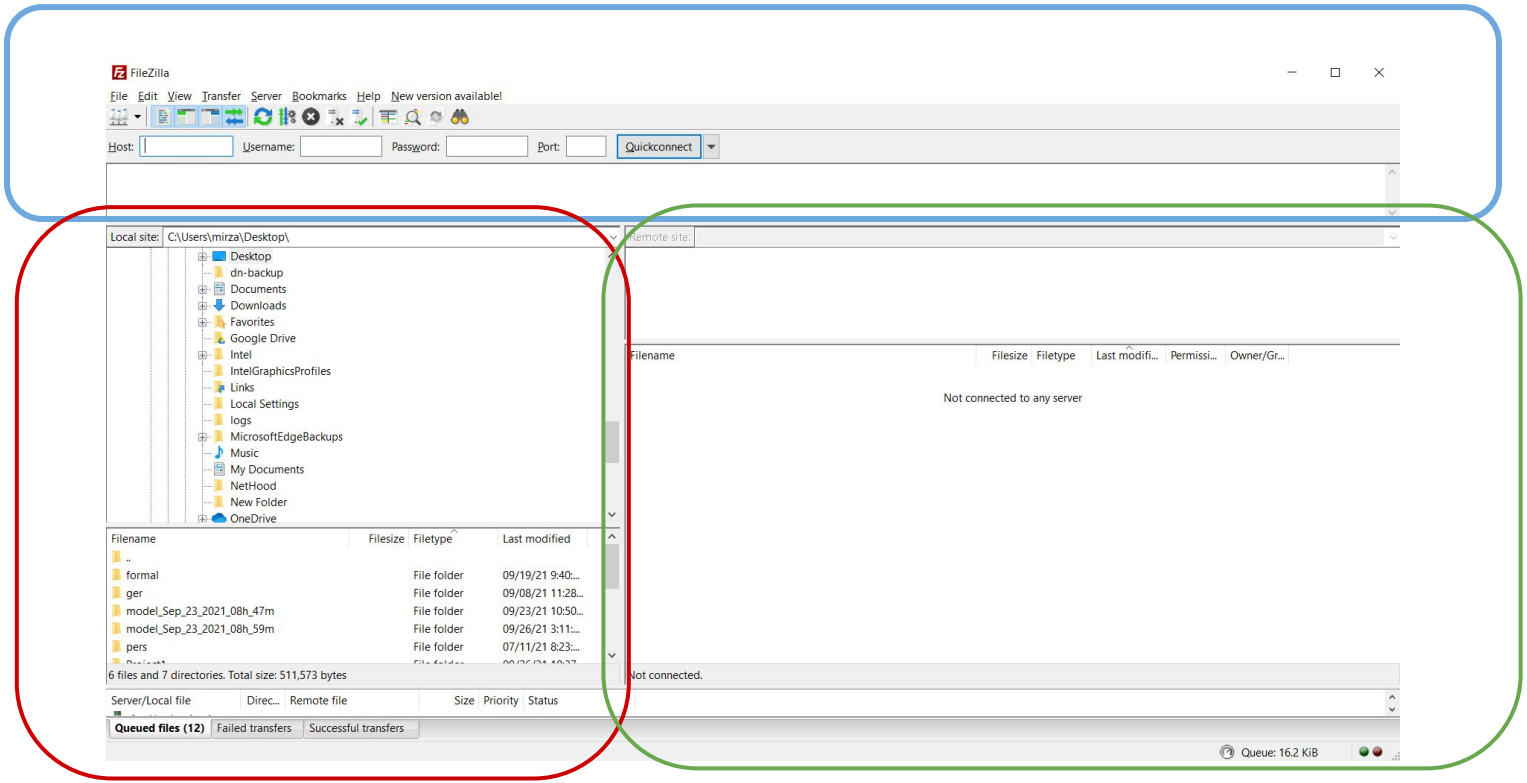
3. Running your code on a remote server

- Before this recitation or shortly after, each student taking this course will receive a unique server ID and password which they will use to connect to the server.

- With that being said. Lets go and upload our code to the server using FileZilla, our ID and password.

3. Running your code on a remote server

Open FileZilla. It has 3 sections. The one marked with blue is for establishing connection, the red section is the contents of your computer and green section is the contents of the server. Because we are not connected yet, the blue and green sections are empty



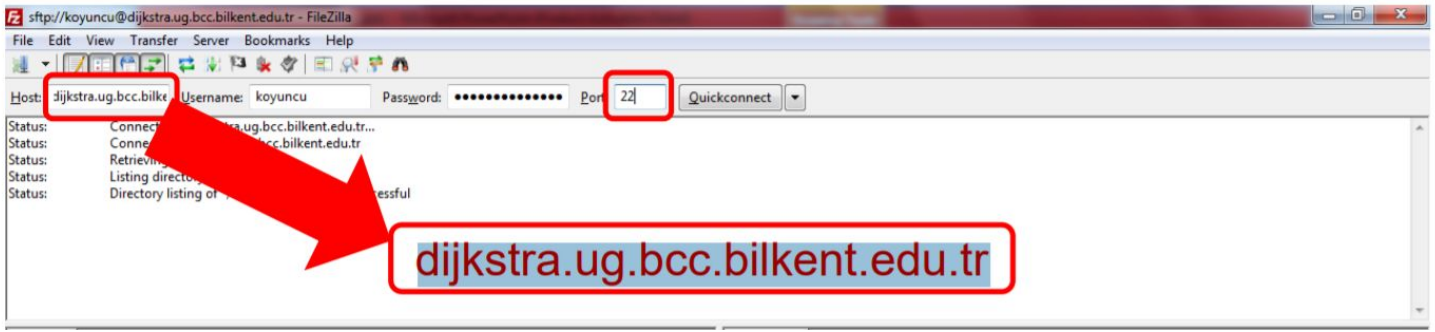
3. Running your code on a remote server

Establish a connection. On the upper corner you will see four slots.

Host: dijkstra.ug.bcc.bilkent.edu.tr
Username: yourusername
Password: yourpassword
Port: 22

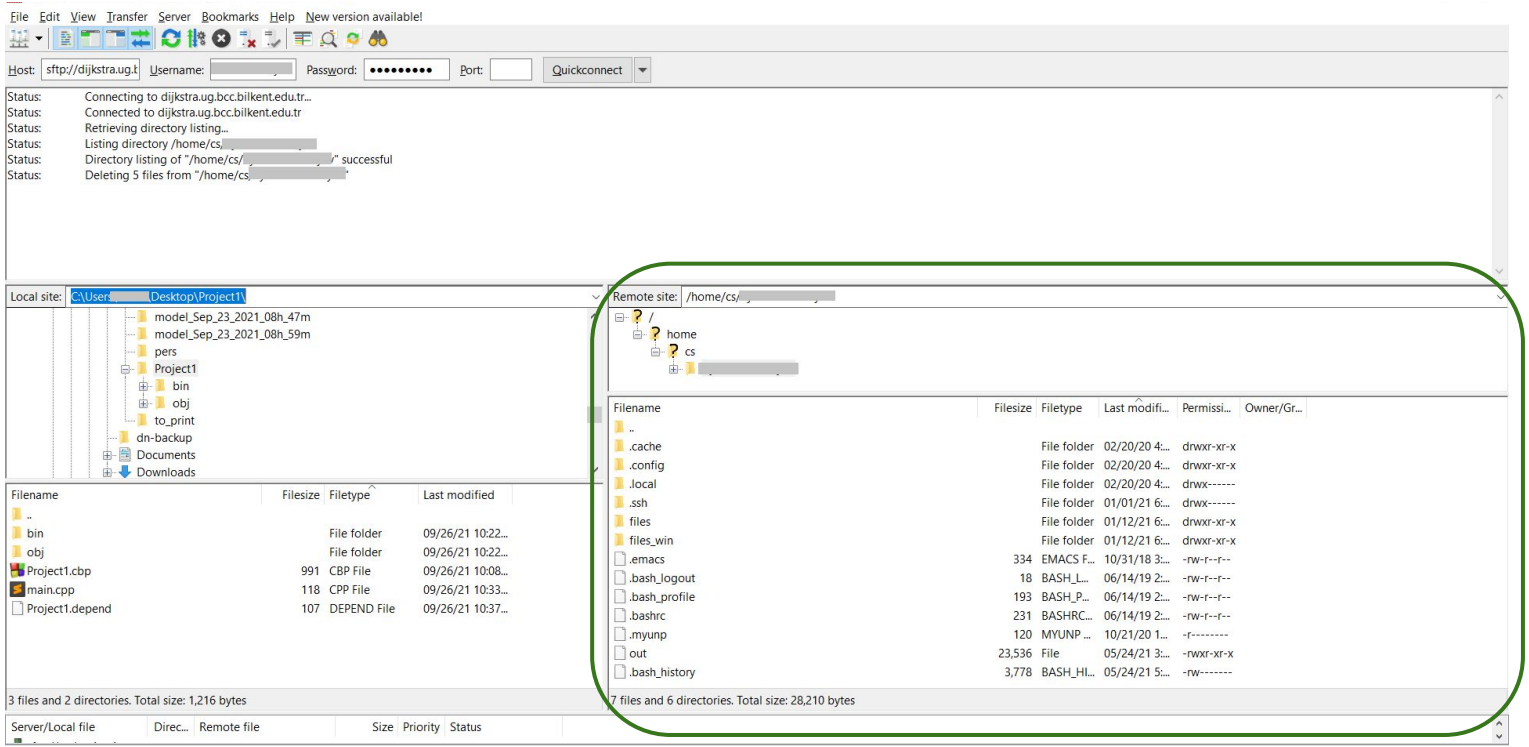
Here host (dijkstra.ug.bcc.bilkent.edu.tr) is the address of our server on the Bilkent network. And username and password are the ID details that each individual student received.

Press Quickconnect to connect to the server.



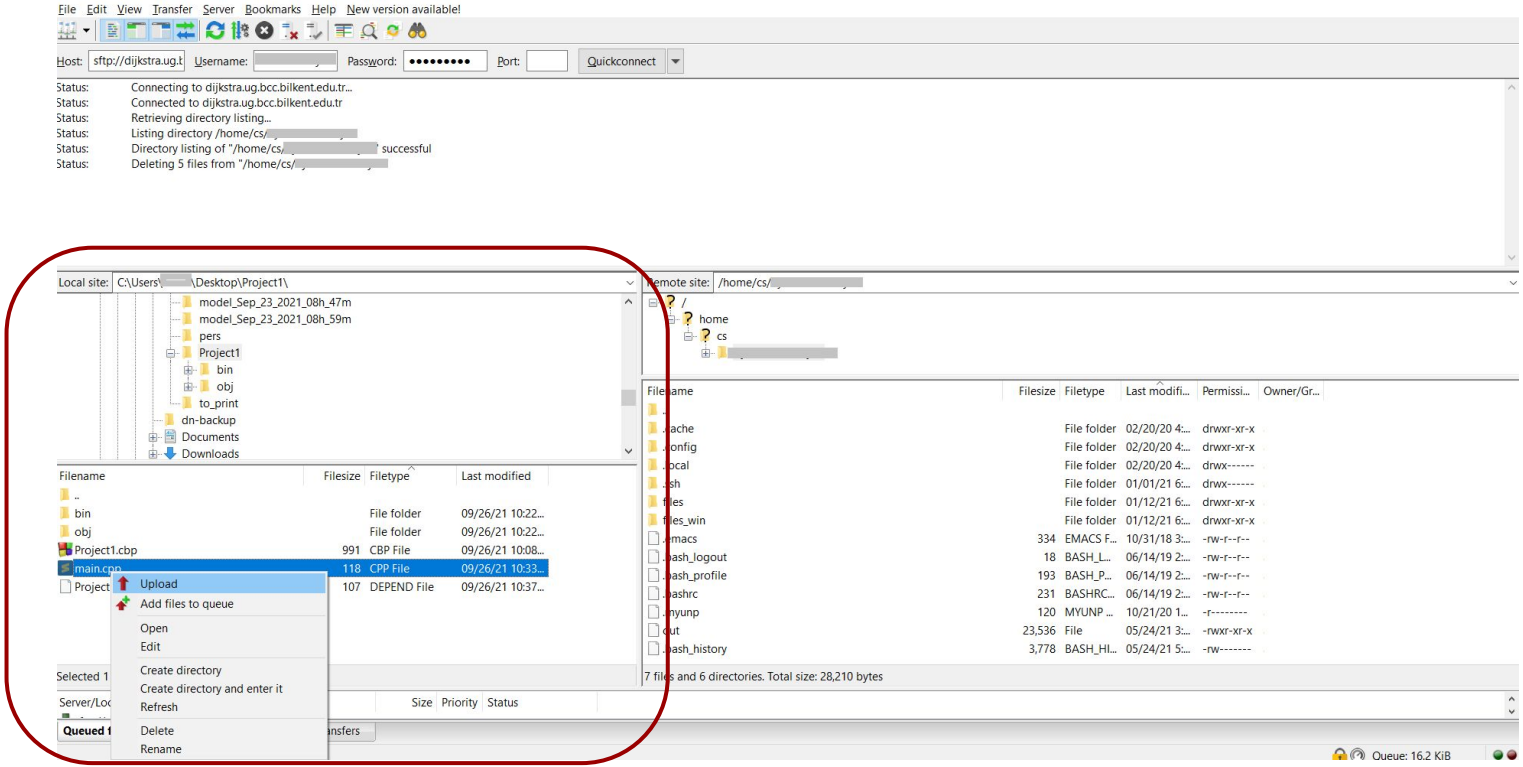
3. Running your code on a remote server

Once connected it will looks something like this. On the right you will see the contents of your server folder. For now it only contains some random files that you don't need to worry about. Let's upload our code.



3. Running your code on a remote server

On the left locate the main.cpp file we previously created, right click and press upload. Once you do that the same file will appear in the window on the right. This means you successfully uploaded your code to the server. You are now ready to compile and run it.

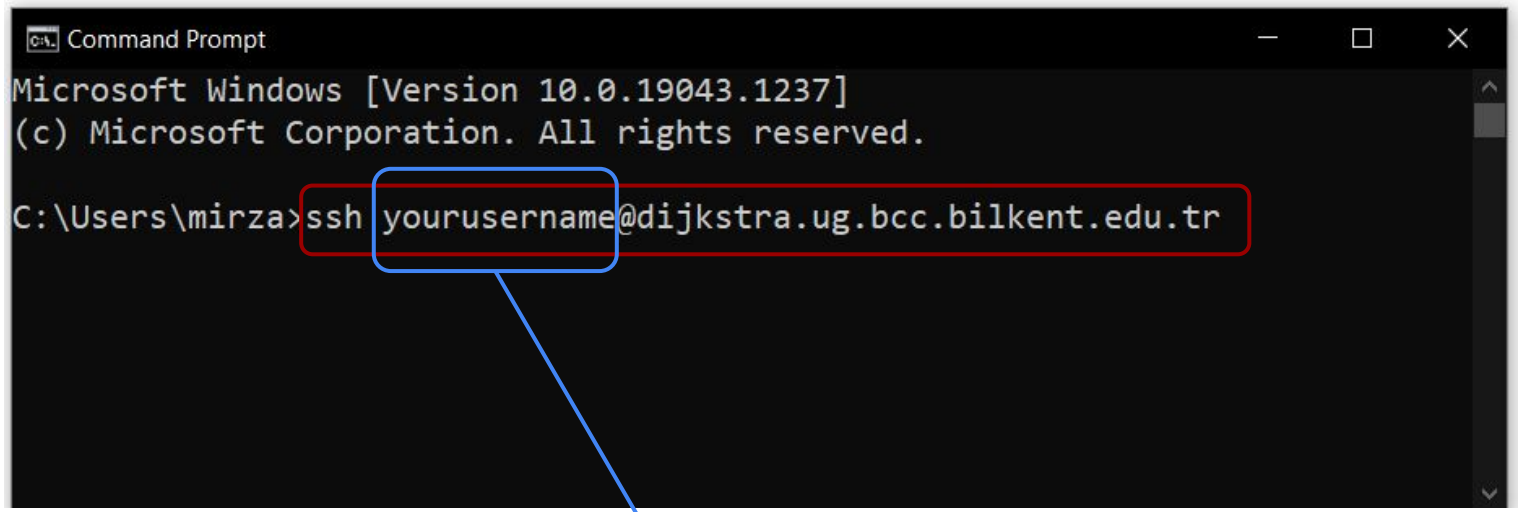


3. Running your code on a remote server

- Next we will run our code on the server
- Some of you may already know that it is possible to run programs from your command prompt just as you do from IDE by pressing run button. Some of you might have already used command prompt to run codes.
- Here we will use our command prompt to connect to the remove server and run codes on the server.

3. Running your code on a remote server

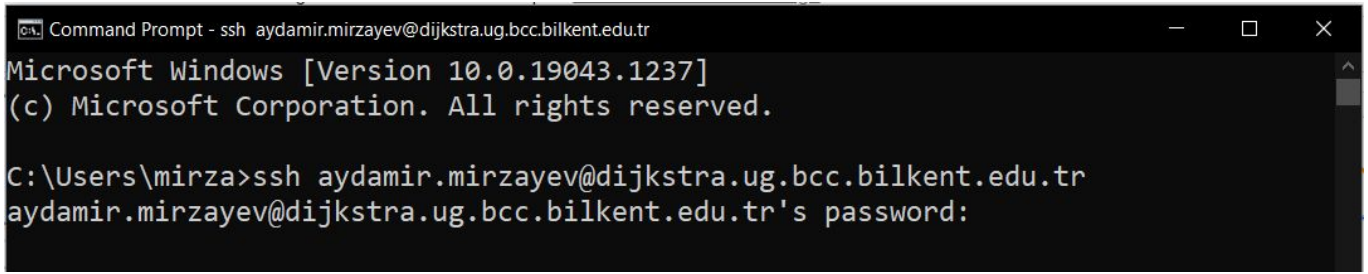
Open your command prompt and type the following command:



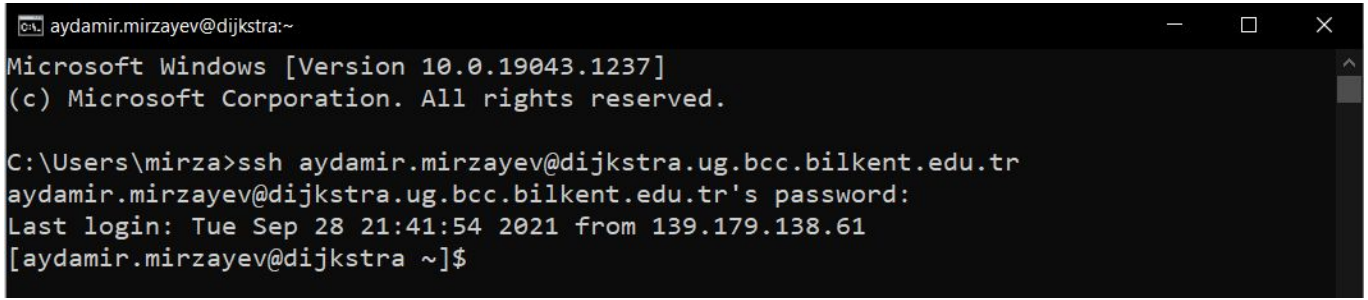
Naturally, you need to replace 'yourusername' part with the username that you have been provided for the server. Type the command and press 'Enter'.

3. Running your code on a remote server

You will be asked to enter a password. As you enter your password you will not see it being typed on the prompt, it is normal, just type the password and press enter.



If you entered it correctly then you will see something like below. Now you command prompt is connected to the server.



3. Running your code on a remote server

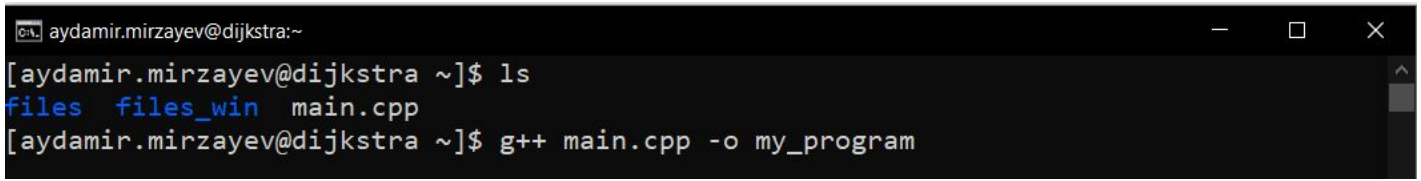
For example if you now type ‘ls’ you will see same files listed in the command prompt.

Here you can also see the main.cpp file that we just uploaded using FileZilla is visible from prompt.
Let’s compile and run it.

To compile the file, you will use command ‘g++ main.cpp -o my_program’

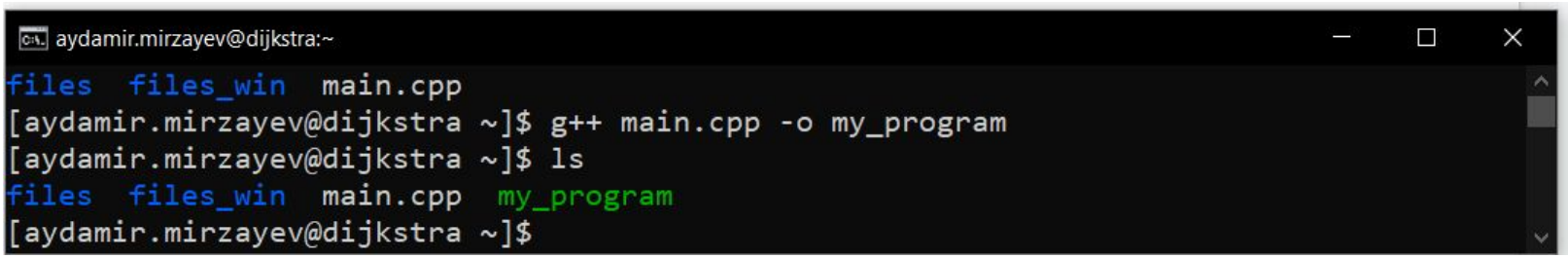
Here: ‘g++’ is the name of the compiler
‘main.cpp’ is the name of the file that we want to compile
‘-o’ is a command for output assignment
‘my_program’ is a name that we choose to give to our compiled binary file

This command will generate a compiled binary named ‘my_program’ that we will use to run the code.

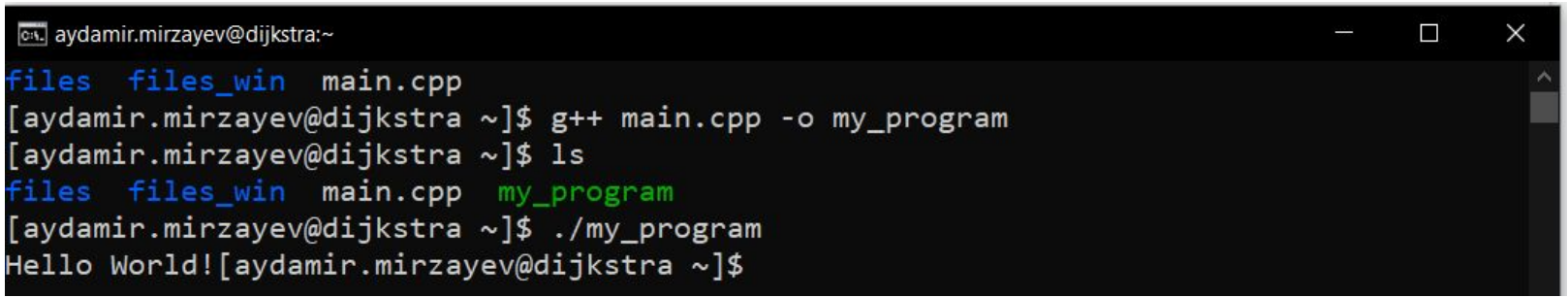


3. Running your code on a remote server

After you execute the compile command you can actually see my_program binary that we just created by listing the folder using 'ls'



Finally to run the binary, type: ./my_program



Hello World! Is printed. Congrats! You ran your code on the server!

3. Running your code on a remote server

- Okay, now we know how to run a single .cpp file on the server. But how do we run large projects?
- It is not much different, the exact same procedure with a minor difference.
- Project might contain header files. We need to include them. What are they?

4. Running a project with multiple header and cpp files.

- Header files are used to separate declaration and implementation in C++. For the sake of this recitation you just need to know that they are part of the project, they are linked with .cpp files, and they need to be present to be able to compile the project.

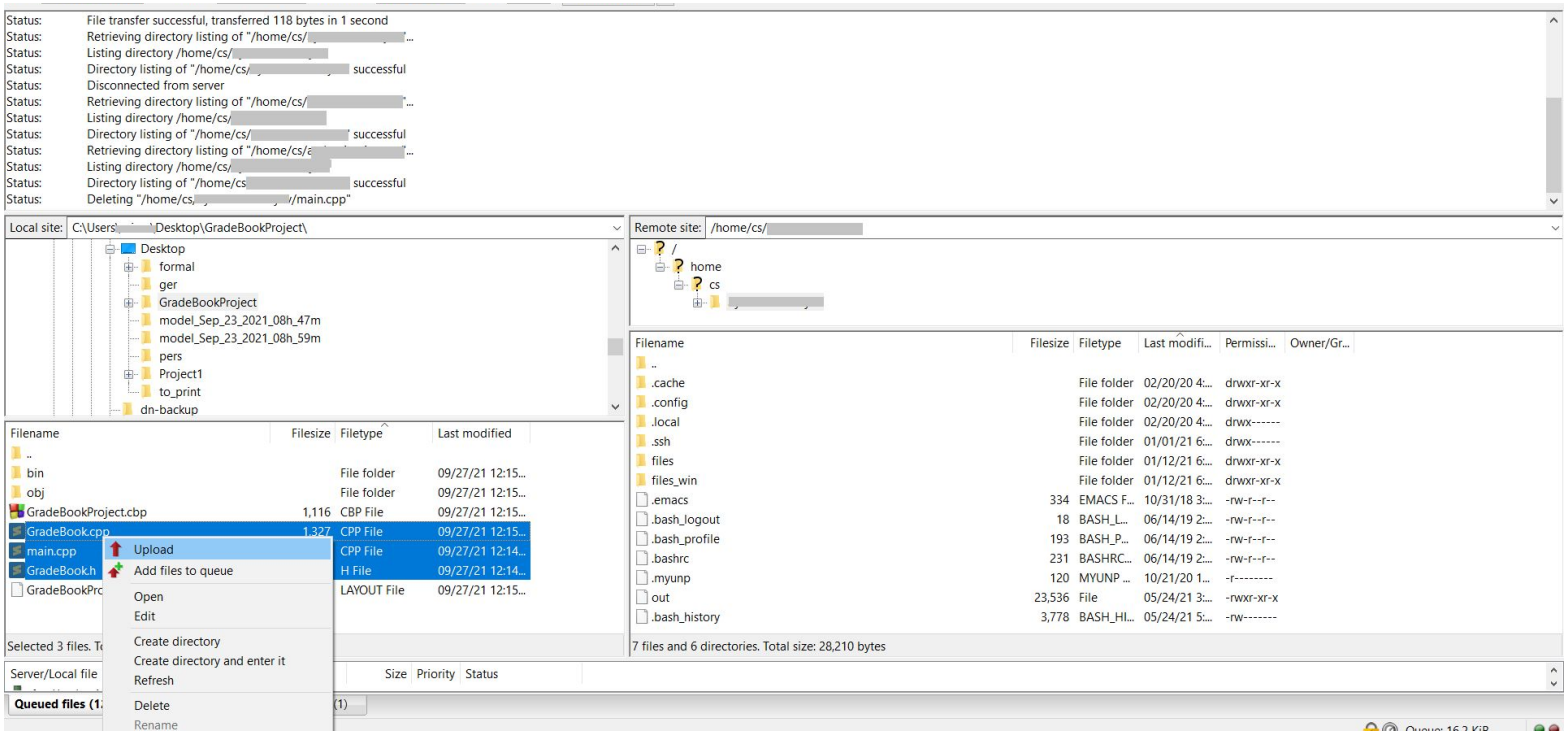
- In this recitation we will use a ready project named GradeBook that contains a header file. You don't need to worry about implementation we will provide you with the code.

- The program is very simple, it asks for grades of the student on individual assignments and prints the letter grade that the student is going to receive from the course.

- The program has 3 files. main.cpp, GradeBook.cpp and GradeBook.h

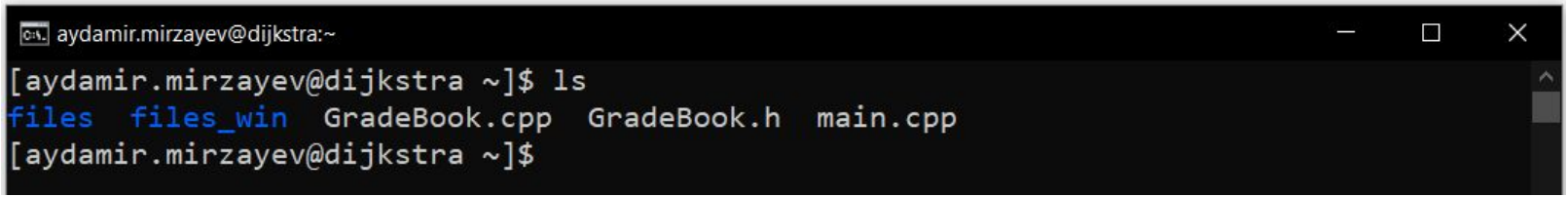
4. Running a project with multiple header and cpp files.

- Locate .h and .cpp files in the project folder and upload them just as we did with single main.cpp file.

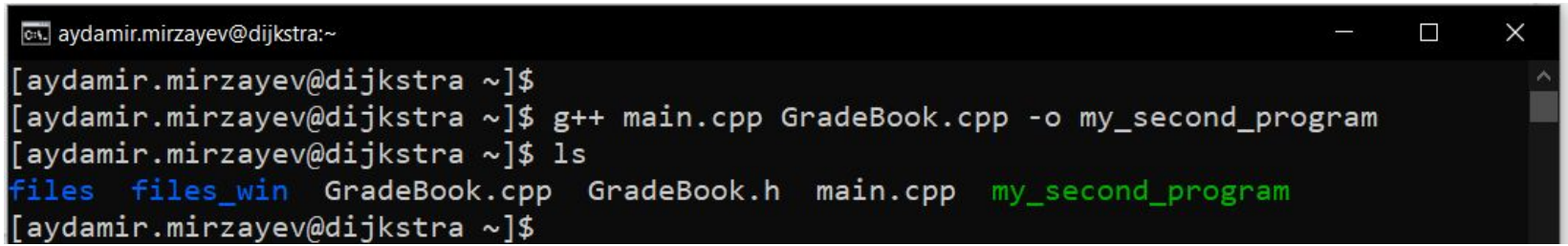


4. Running a project with multiple header and cpp files.

Now, navigate to the command prompt again. Run 'ls' command again, you should be able to see the files that you have uploaded.

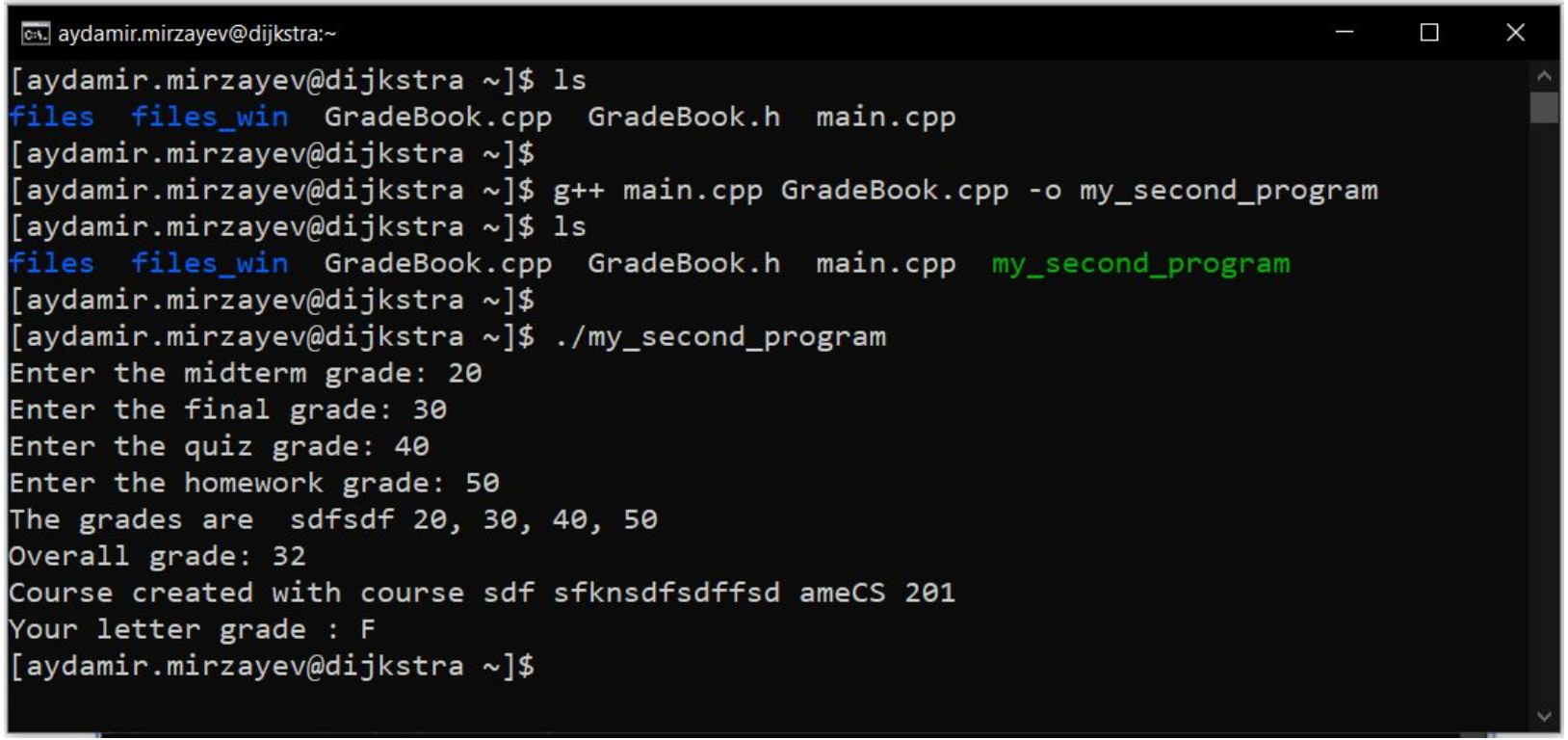


Now run: g++ main.cpp GradeBook.cpp -o my_second_program to generate the binary of this program. As you might have guessed, when we are compiling a project with multiple cpp files, we simply type the names of all cpp files instead of a single one. You might have also noticed that we don't type the name of .h file. This is because .h files are internally linked with .cpp files and don't need to be included in the command. But they need to be present in the same folder.



4. Running a project with multiple header and cpp files.

Now you simply run you new binary.



4. Running a project with multiple header and cpp files.

- That's it. If you reached this point. You are ready to test your code on the server before submitting it.
- Always make sure that you test your code on the server before you submit it
- Always make sure that you are ONLY using 'g++' compiler.
- Uploading you code to the server is not same as submitting it. Your code on the server is seen by you and only you. You still need to upload your code to the Moodle. And that is what we will use for grading.
- Next I will discuss an extra trick that might make your life a bit easier. It is called Makefile.

5. Checking memory leaks

- On almost all of your assignments for CS201 course, you will be required to check the program that you are submitting for memory leaks.
- You can use an extension called Valgrind
- You can either install it on your computer on choose to use the server to check your code for leaks
- Here we will see how to check for memory leaks on the server using Valgrind

5. Checking memory leaks

- Assume we have a program that is composed of files named: Cabinet.h, Chemical.h, LabOrganizer.h, Chemical.cpp, Cabinet.cpp, LabOrganizer.cpp, and main.cpp

- Let us check this program for memory leaks on the server using Valgrind
- We run the command g++ Cabinet.cpp Chemical.cpp LabOrganizer.cpp main.cpp -g -o my_program to compile the program

```
[aydamir.mirzayev@dijkstra ~]$ ls
Cabinet.cpp  Cabinet.h  Chemical.cpp  Chemical.h  files_win  LabOrganizer.cpp  LabOrganizer.h  main.cpp
[aydamir.mirzayev@dijkstra ~]$ g++ Cabinet.cpp Chemical.cpp LabOrganizer.cpp main.cpp -g -o my_program
```

- Here, we use the additional -g command to enable the debug mode which in turn will provide us with more control over the binary. This command is not compulsory but it is nice to include.

5. Checking memory leaks

- Once we compiled the binary, we use the command valgrind --tool=memcheck --leak-check=yes ./my_program

```
[aydamir.mirzayev@dijkstra ~]$ ls
Cabinet.cpp  Cabinet.h  Chemical.cpp  Chemical.h  files_win  LabOrganizer.cpp  LabOrganizer.h  main.cpp
[aydamir.mirzayev@dijkstra ~]$ g++ Cabinet.cpp Chemical.cpp LabOrganizer.cpp main.cpp -g -o my_program
[aydamir.mirzayev@dijkstra ~]$ valgrind --tool=memcheck --leak-check=yes ./my_program
```

- If at the end of your program you see and output like the one below mentioning that no leaks are possible, then your program does not have memory leaks. For more information refer to online resources on Valgrind

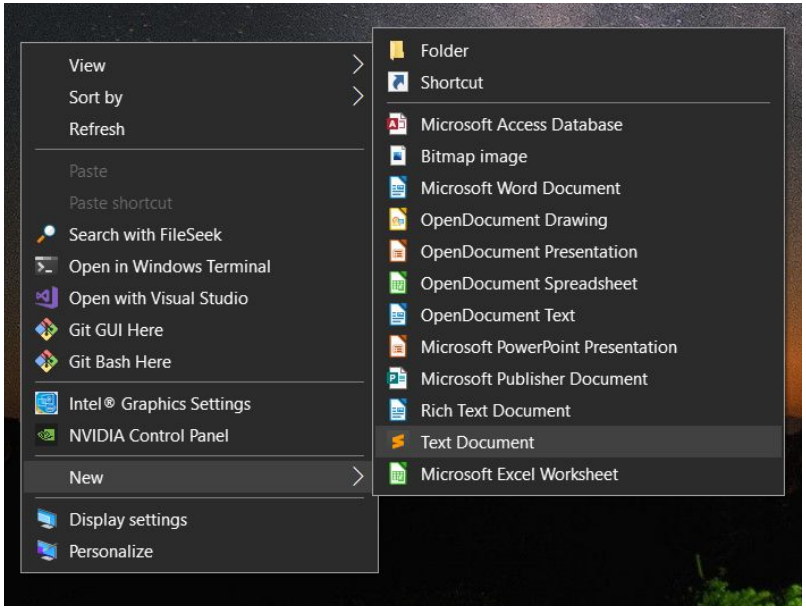
```
==16808==
==16808== HEAP SUMMARY:
==16808==     in use at exit: 0 bytes in 0 blocks
==16808==   total heap usage: 448 allocs, 448 frees, 22,626 bytes allocated
==16808==
==16808== All heap blocks were freed -- no leaks are possible
==16808==
==16808== For lists of detected and suppressed errors, rerun with: -s
==16808== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[aydamir.mirzayev@dijkstra ~]$
```

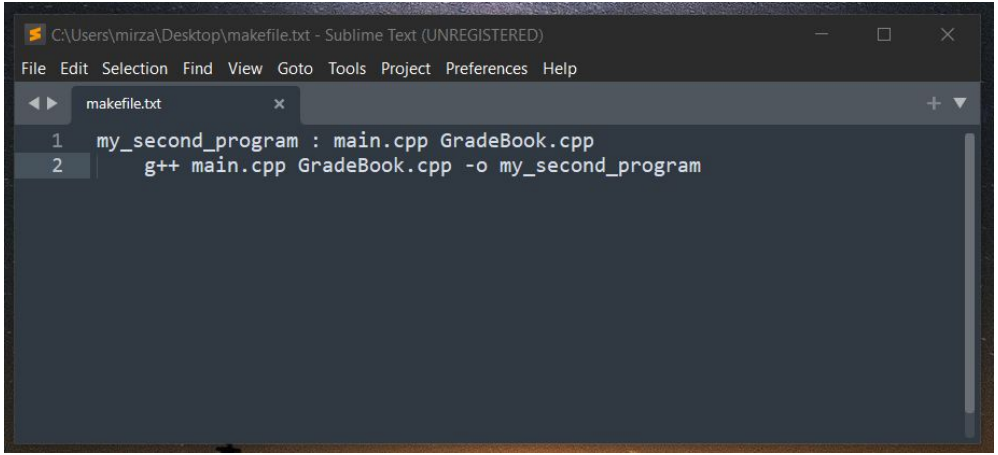
6. Briefly about Makefile

- Makefile is often used for making compiling large projects a bit easier.
- You now know that we need to type the names of all of our cpp files when we compile a project on the server.
- You can imagine how on a project with 10-15 cpp files compile command might get messy: g++ main.cpp car.cpp house.cpp person.cpp ...
- Makefile comes in handy in similar situations.
- Best way to explain this is to demonstrate and example. So let's do it.

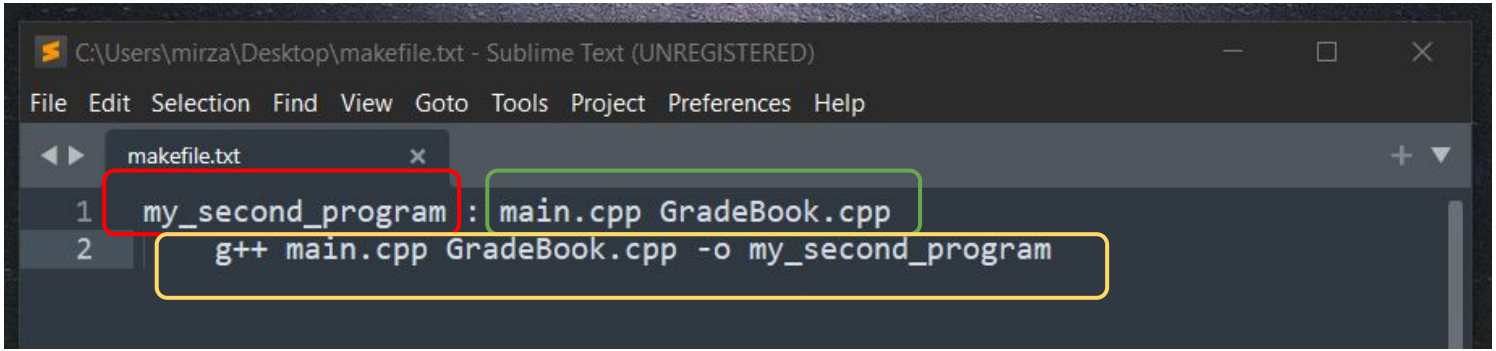
6. Briefly about Makefile

On your computer create a txt file called ‘makefile.txt’. Open that txt file and type the following:





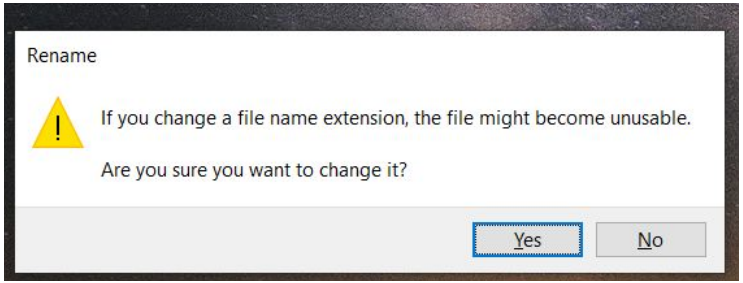
6. Briefly about Makefile

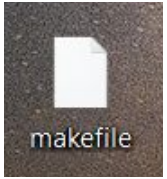


- Red section on the left is the name of your output
- Green section on the right contains the names of the files you will use to generate the output
- And the yellow section at the bottom is the really long command that you don't want to type over and over again on the command prompt

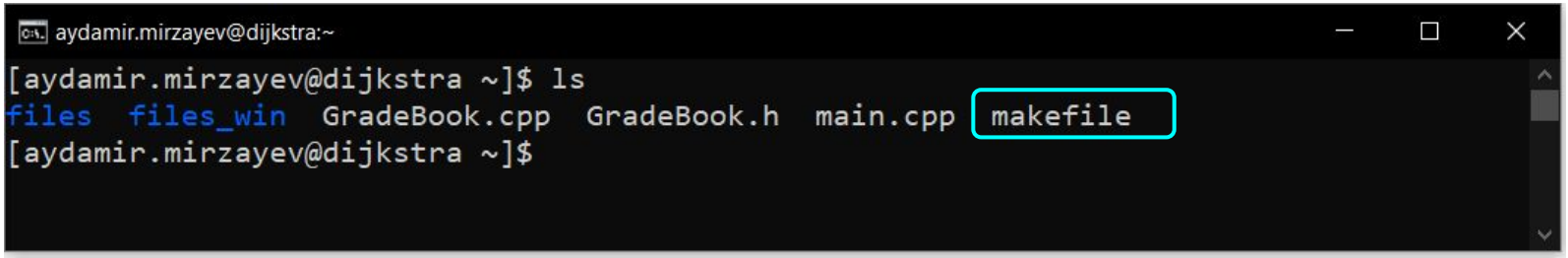
6. Briefly about Makefile

Save and close the file. After you closed it rename it such that it no longer has the .txt extension.



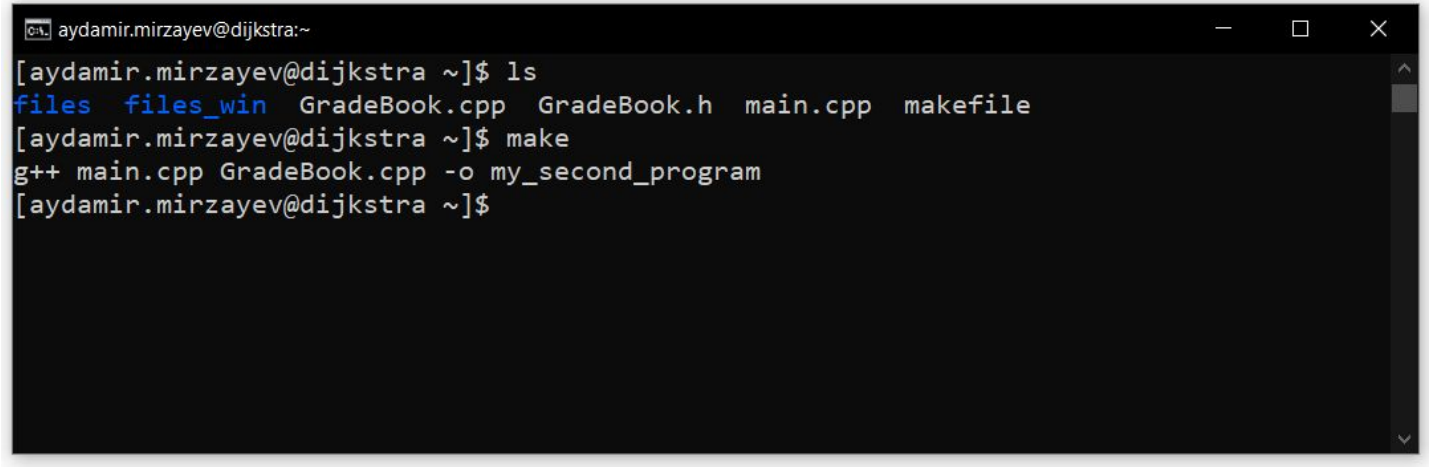


You will get a warning. Just press yes. Now upload this file to the server just as you uploaded your code. After you upload the makefile go to the command prompt and type 'ls' to observe the makefile in the directory.



6. Briefly about Makefile

Now, to execute the same ‘g++ main.cpp Gradebook.cpp -o my_second_output’ command you simply need to type ’make’.



And now your long command has been executed by just typing ‘make’

6. Briefly about Makefile

- In all honesty though, Makefile is a very powerful tool used by professional teams. It does more than just simplify commands. It optimizes the compilation process as well.

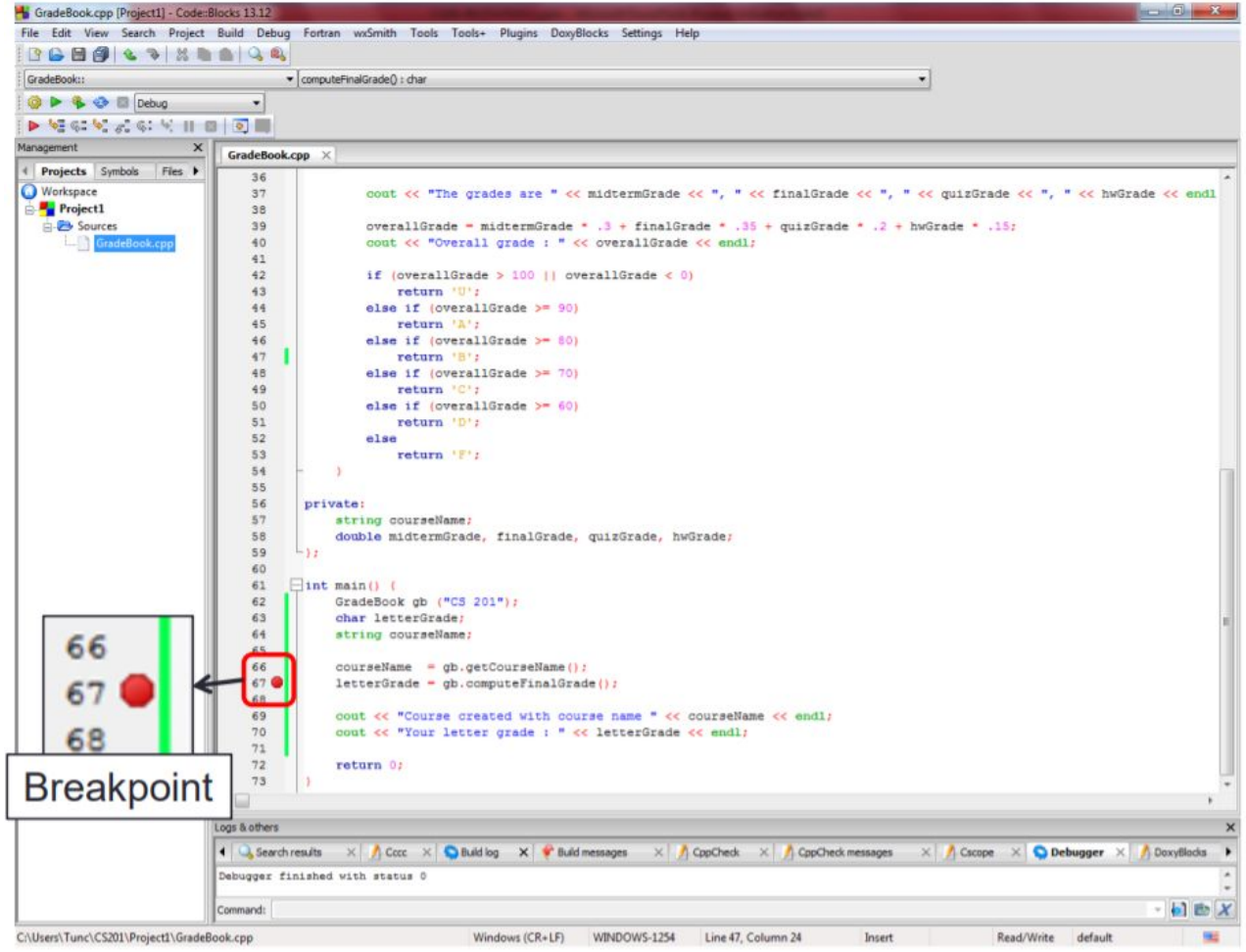
- But that is outside the scope of this course. But we do encourage you to read up on the Makefile.

- Please refer to https://makefiletutorial.com/ for more information on usage of Makefile

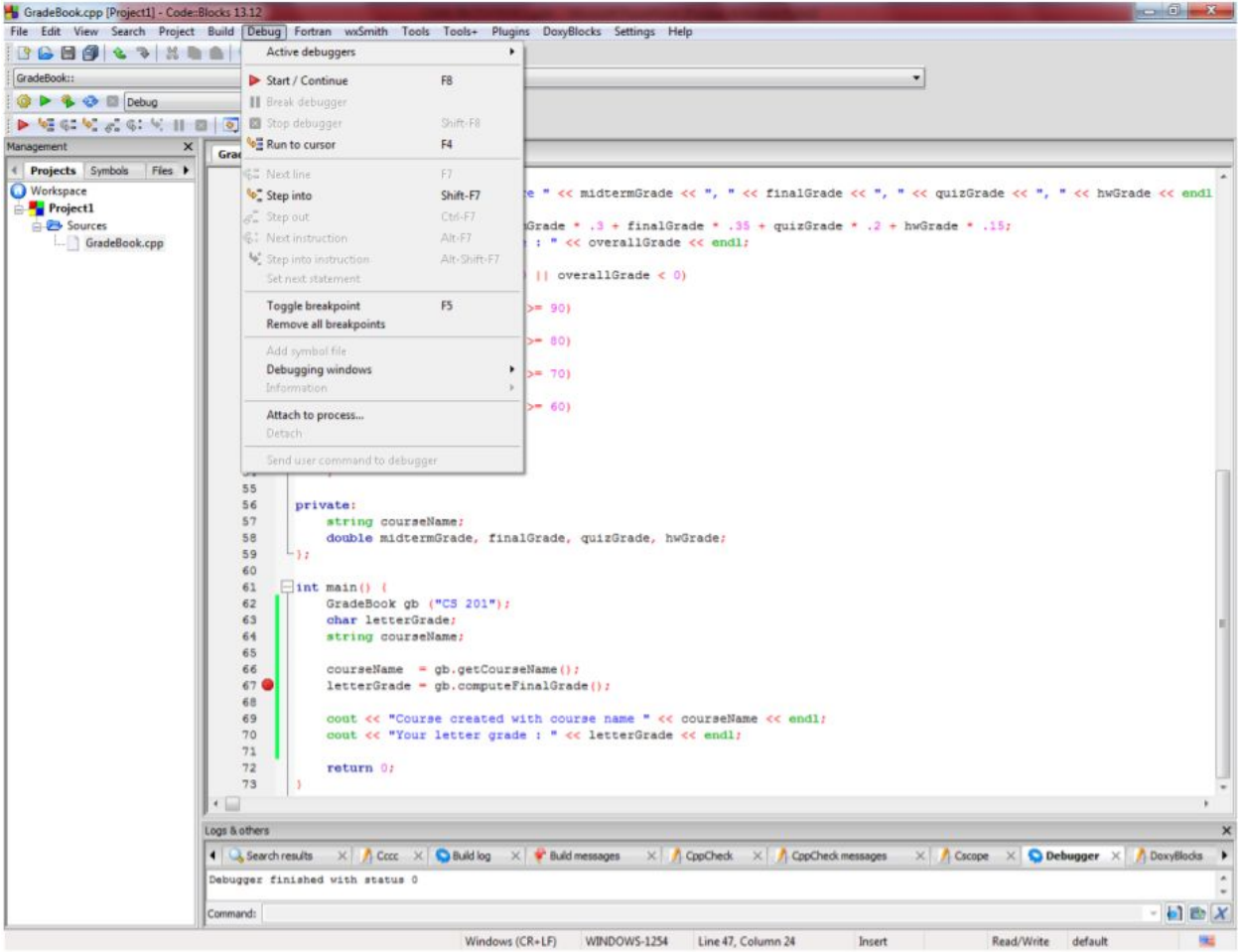
7. Debugging

- In debugging mode you can track your code as it is being executed
- Instead of running the entire code at once you can watch step-by-step as each command is being executed
- Let's open the project GradeBook and try each step

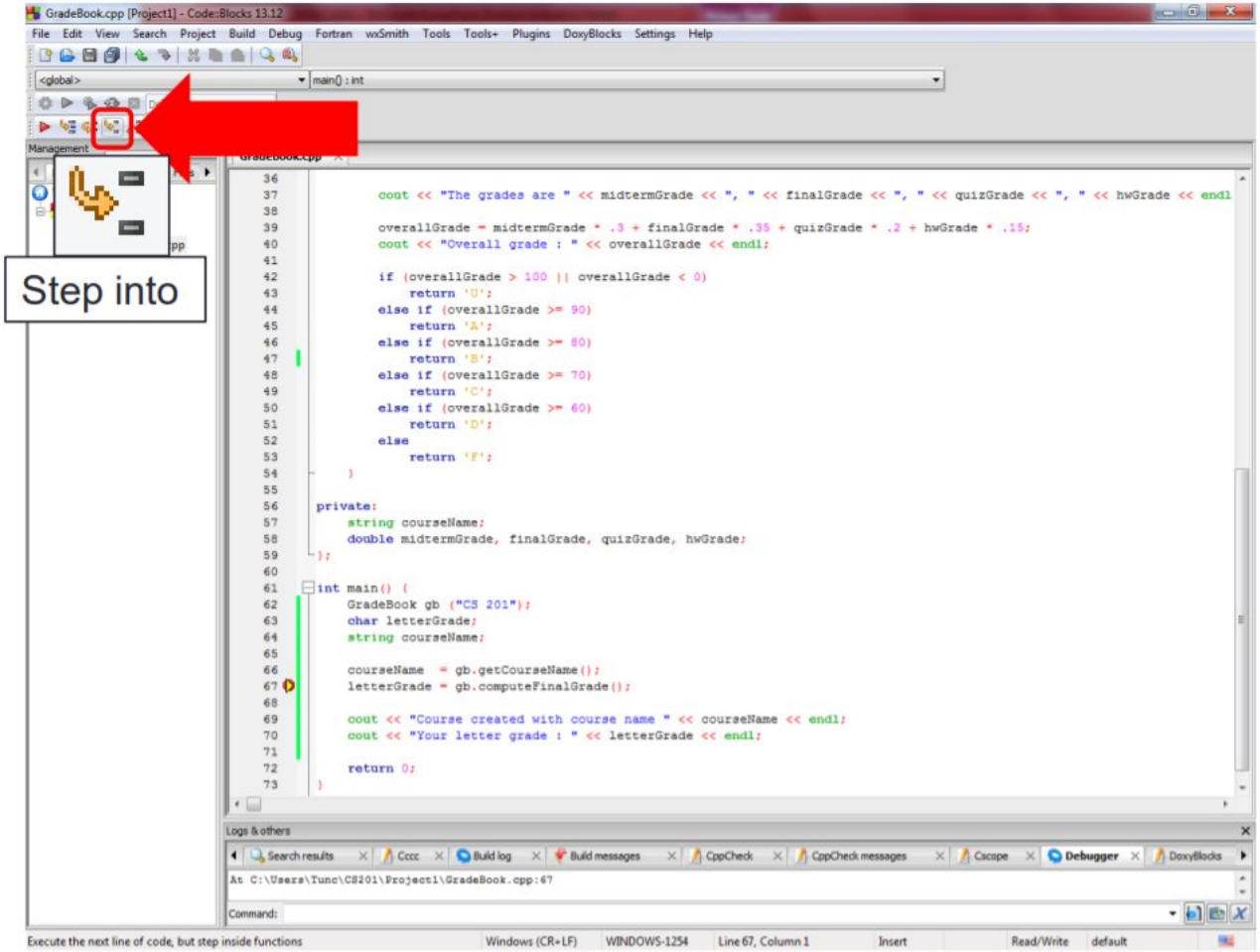
7. Debugging


GradeBook.cpp [Project1] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
Project1
Sources
GradeBook.cpp
GradeBook.cpp
cout << "The grades are " << midtermGrade << ", " << finalGrade << ", " << quizGrade << ", " << hwGrade << endl
overallGrade = midtermGrade * .3 + finalGrade * .35 + quizGrade * .2 + hwGrade * .15;
cout << "Overall grade : " << overallGrade << endl;
if (overallGrade > 100 || overallGrade < 0)
return 'U';
else if (overallGrade >= 90)
return 'A';
else if (overallGrade >= 80)
return 'B';
else if (overallGrade >= 70)
return 'C';
else if (overallGrade >= 60)
return 'D';
else
return 'F';
}
private:
string courseName;
double midtermGrade, finalGrade, quizGrade, hwGrade;
};
int main() {
GradeBook gb ("CS 201");
char letterGrade;
string courseName;
courseName = gb.getCourseName();
letterGrade = gb.computeFinalGrade();
cout << "Course created with course name " << courseName << endl;
cout << "Your letter grade : " << letterGrade << endl;
return 0;
}
66
67
68
Breakpoint
Logs & others
Search results Cccc Build log Build messages CppCheck CppCheck messages Cscope Debugger DoxyBlocks
Debugger finished with status 0
Command:
C:\Users\Tunc\CS201\Project1\GradeBook.cpp
Windows (CR+LF)
WINDOWS-1254
Line 47, Column 24
Insert
Read/Write
default

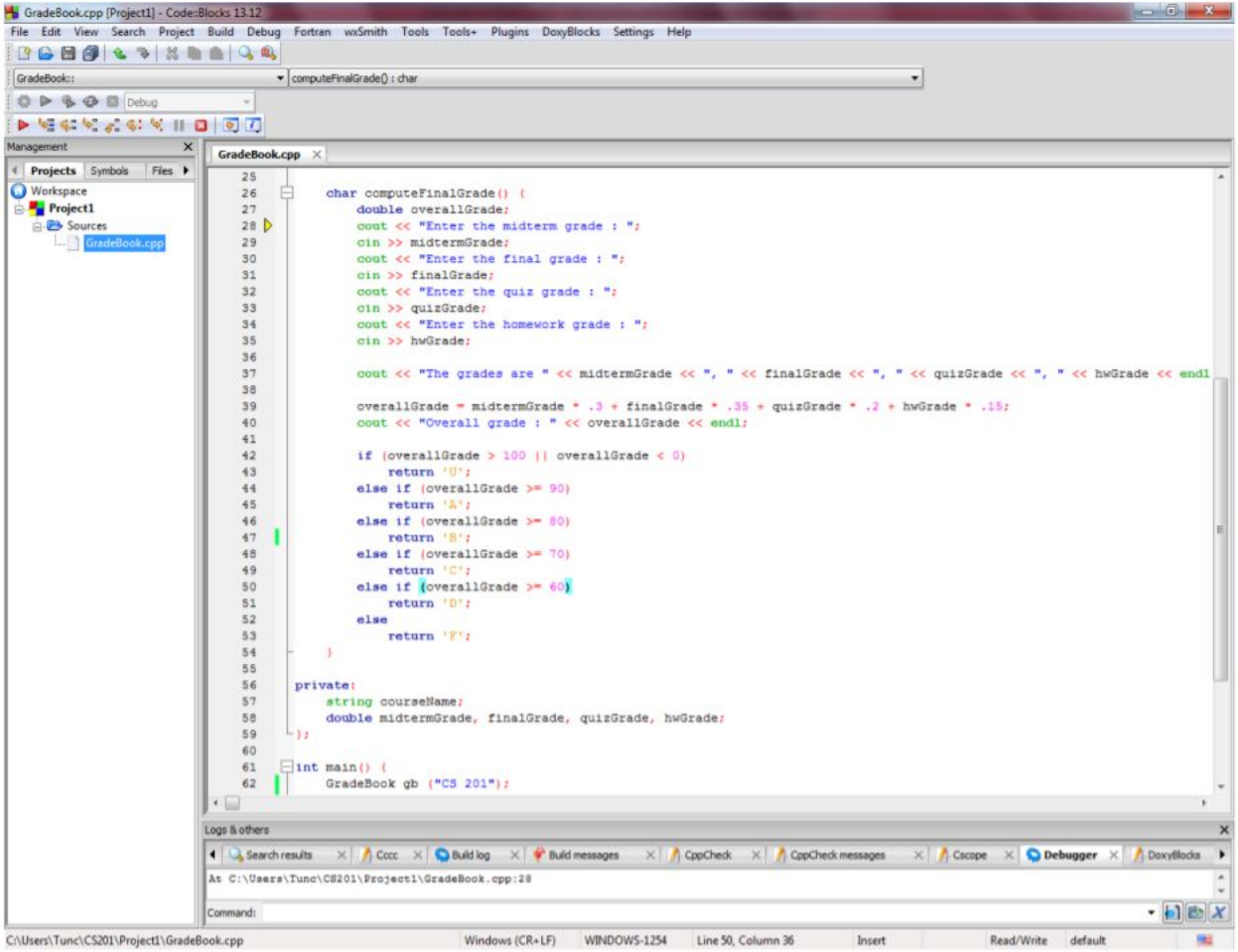
7. Debugging


GradeBook.cpp [Project1] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Active debuggers
Start / Continue F8
Break debugger
Stop debugger Shift-F8
Run to cursor F4
Next line F7
Step into Shift-F7
Step out Ctrl-F7
Next instruction Alt-F7
Step into instruction Alt-Shift-F7
Set next statement
Toggle breakpoint F5
Remove all breakpoints
Add symbol file
Debugging windows
Information
Attach to process...
Detach
Send user command to debugger
Management
Projects Symbols Files
Workspace
Project1
Sources
GradeBook.cpp
private:
string courseName;
double midtermGrade, finalGrade, quizGrade, hwGrade;
};
int main() {
GradeBook gb ("CS 201");
char letterGrade;
string courseName;
courseName = gb.getCourseName();
letterGrade = gb.computeFinalGrade();
cout << "Course created with course name " << courseName << endl;
cout << "Your letter grade : " << letterGrade << endl;
return 0;
}
Logs & others
Search results Cccc Build log Build messages CppCheck CppCheck messages Cscope Debugger DoxyBlocks
Debugger finished with status 0
Command:
Windows (CR+LF) WINDOWS-1254 Line 47, Column 24 Insert Read/Write default

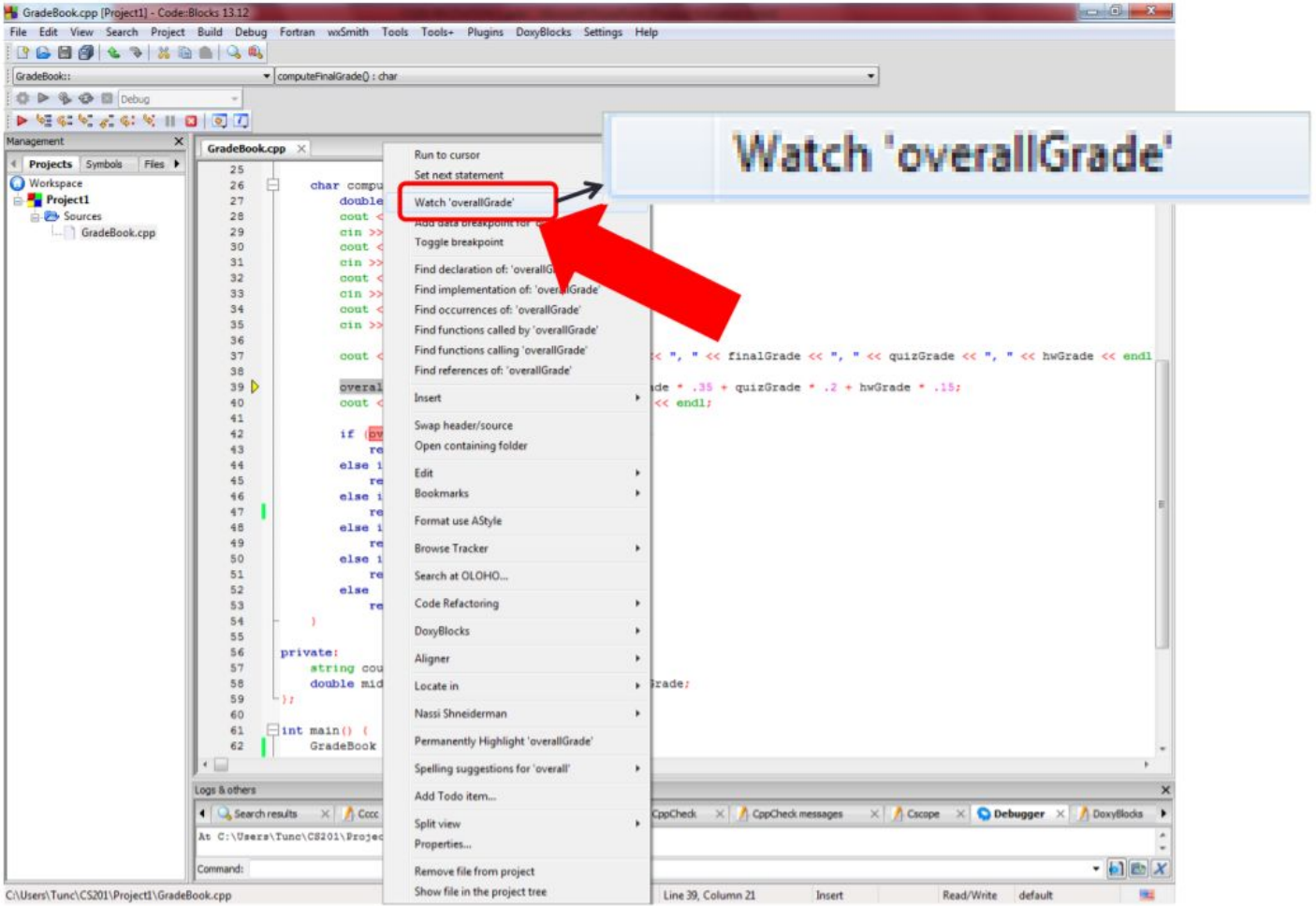
7. Debugging


GradeBook.cpp [Project1] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main() : int
Management
Step into
cout << "The grades are " << midtermGrade << ", " << finalGrade << ", " << quizGrade << ", " << hwGrade << endl
overallGrade = midtermGrade * .3 + finalGrade * .35 + quizGrade * .2 + hwGrade * .15;
cout << "Overall grade : " << overallGrade << endl;
if (overallGrade > 100 || overallGrade < 0)
return 'U';
else if (overallGrade >= 90)
return 'A';
else if (overallGrade >= 80)
return 'B';
else if (overallGrade >= 70)
return 'C';
else if (overallGrade >= 60)
return 'D';
else
return 'F';
}
private:
string courseName;
double midtermGrade, finalGrade, quizGrade, hwGrade;
};
int main() {
GradeBook gb ("CS 201");
char letterGrade;
string courseName;
courseName = gb.getCourseName();
letterGrade = gb.computeFinalGrade();
cout << "Course created with course name " << courseName << endl;
cout << "Your letter grade : " << letterGrade << endl;
return 0;
}
Logs & others
Search results
Cccc
Build log
Build messages
CppCheck
CppCheck messages
Cscope
Debugger
DoxyBlocks
At C:\Users\Tunc\CS201\Project1\GradeBook.cpp:67
Command:
Execute the next line of code, but step inside functions
Windows (CR+LF)
WINDOWS-1254
Line 67, Column 1
Insert
Read/Write
default

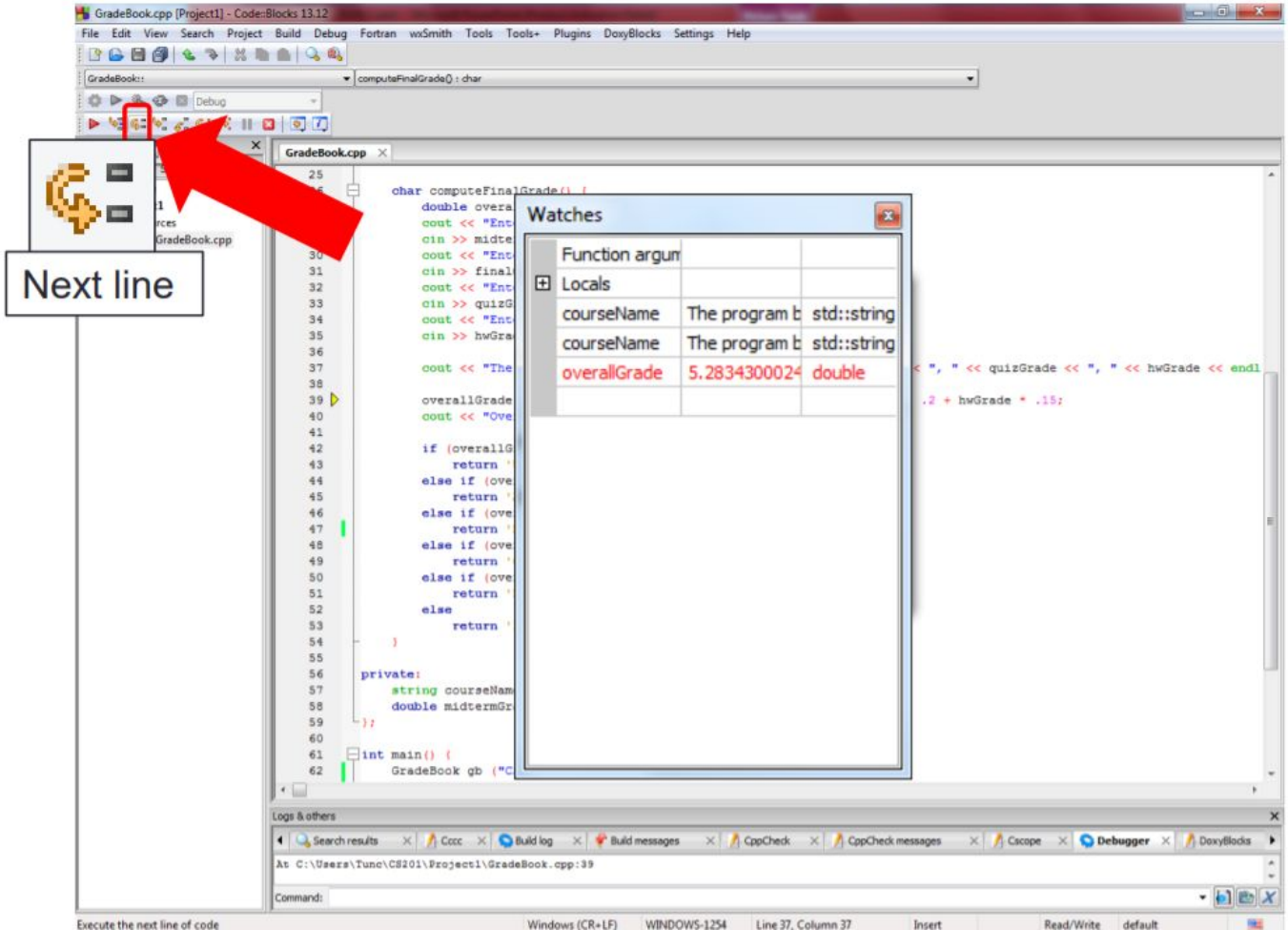
7. Debugging


GradeBook.cpp [Project1] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
GradeBook::
computeFinalGrade() : char
Debug
Management
Projects Symbols Files
Workspace
Project1
Sources
GradeBook.cpp
GradeBook.cpp
25
26 char computeFinalGrade() {
27 double overallGrade;
28 cout << "Enter the midterm grade : ";
29 cin >> midtermGrade;
30 cout << "Enter the final grade : ";
31 cin >> finalGrade;
32 cout << "Enter the quiz grade : ";
33 cin >> quizGrade;
34 cout << "Enter the homework grade : ";
35 cin >> hwGrade;
36
37 cout << "The grades are " << midtermGrade << ", " << finalGrade << ", " << quizGrade << ", " << hwGrade << endl
38
39 overallGrade = midtermGrade * .3 + finalGrade * .35 + quizGrade * .2 + hwGrade * .15;
40 cout << "Overall grade : " << overallGrade << endl;
41
42 if (overallGrade > 100 || overallGrade < 0)
43 return 'U';
44 else if (overallGrade >= 90)
45 return 'A';
46 else if (overallGrade >= 80)
47 return 'B';
48 else if (overallGrade >= 70)
49 return 'C';
50 else if (overallGrade >= 60)
51 return 'D';
52 else
53 return 'F';
54 }
55
56 private:
57 string courseName;
58 double midtermGrade, finalGrade, quizGrade, hwGrade;
59 };
60
61 int main() {
62 GradeBook gb ("CS 201");
Logs & others
Search results
Cccc
Build log
Build messages
CppCheck
CppCheck messages
Cscope
Debugger
DoxyBlocks
At C:\Users\Tunc\CS201\Project1\GradeBook.cpp:28
Command:
C:\Users\Tunc\CS201\Project1\GradeBook.cpp
Windows (CR+LF)
WINDOWS-1254
Line 50, Column 36
Insert
Read/Write
default

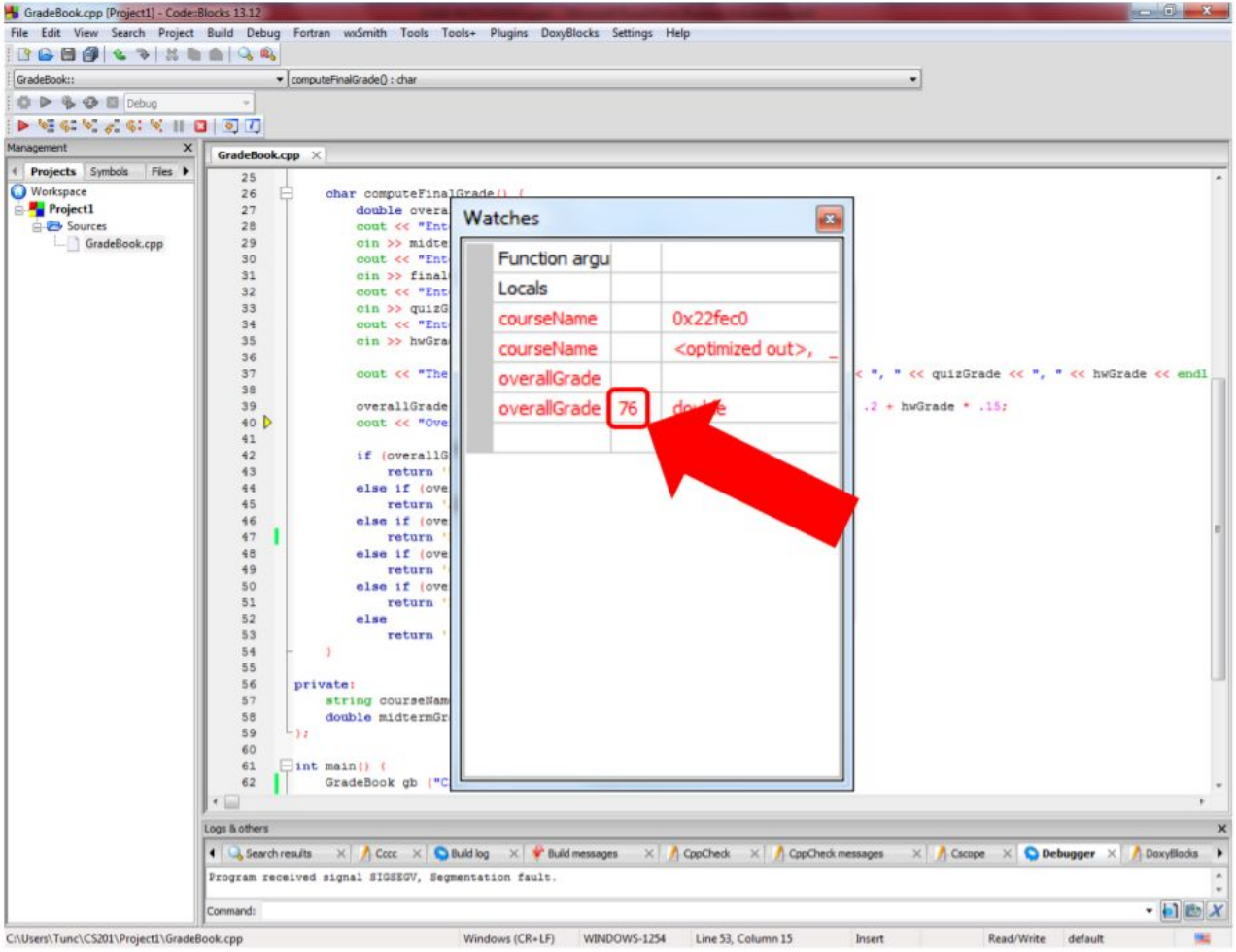
7. Debugging


Watch 'overallGrade'
Run to cursor
Set next statement
Watch 'overallGrade'
Toggle breakpoint
Find declaration of: 'overallGrade'
Find implementation of: 'overallGrade'
Find occurrences of: 'overallGrade'
Find functions called by 'overallGrade'
Find functions calling 'overallGrade'
Find references of: 'overallGrade'
Insert
Swap header/source
Open containing folder
Edit
Bookmarks
Format use AStyle
Browse Tracker
Search at OLOHO...
Code Refactoring
DoxyBlocks
Aligner
Locate in
Nassi Shneiderman
Permanently Highlight 'overallGrade'
Spelling suggestions for 'overall'
Add Todo item...
Split view
Properties...
Remove file from project
Show file in the project tree

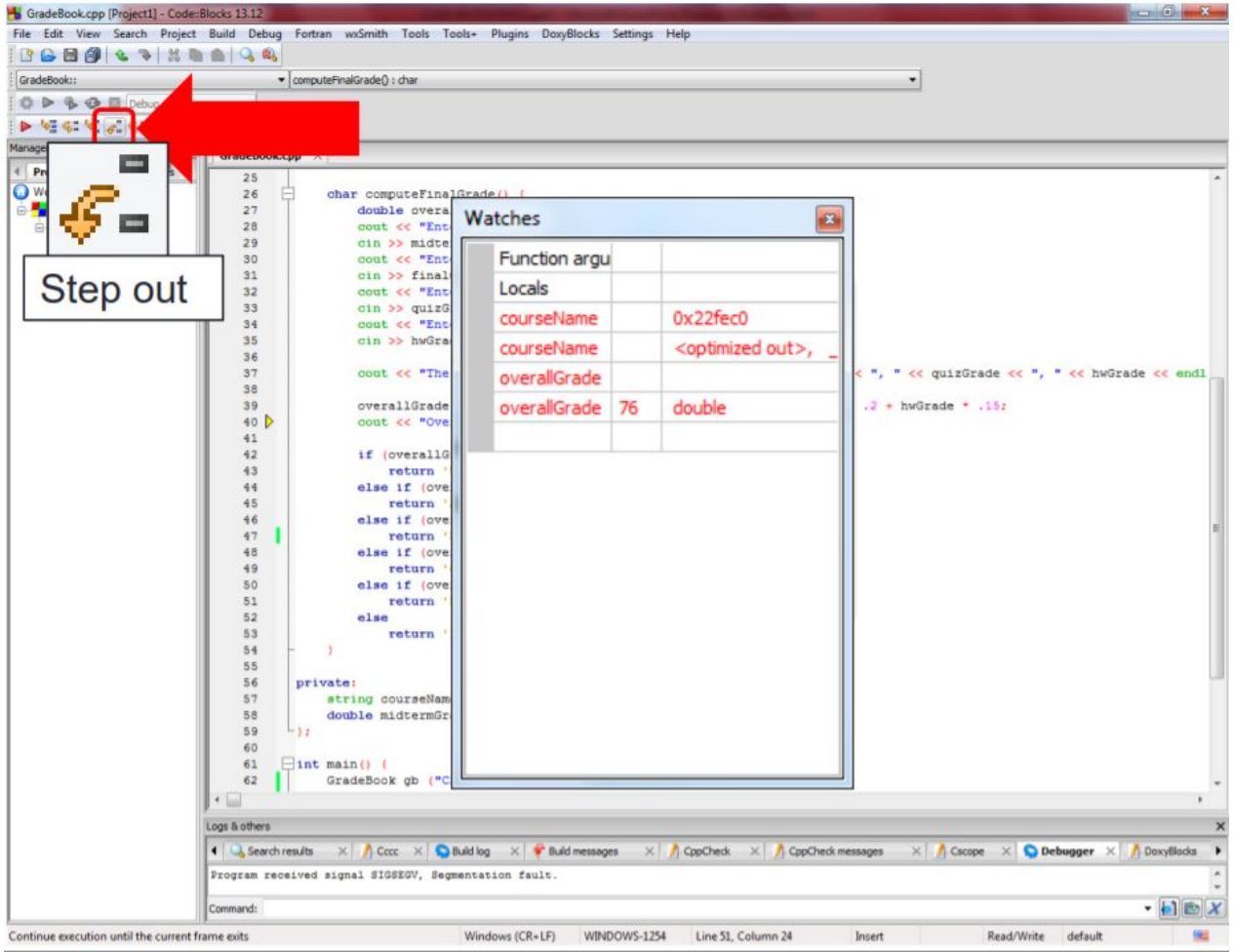
7. Debugging


GradeBook.cpp [Project1] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Next line
Watches
Function argum
Locals
courseName The program b std::string
courseName The program b std::string
overallGrade 5.2834300024 double
Logs & others
Search results Cccc Build log Build messages CppCheck CppCheck messages Cscope Debugger DoxyBlocks
Command:
Execute the next line of code
Windows (CR+LF) WINDOWS-1254 Line 37, Column 37 Insert Read/Write default

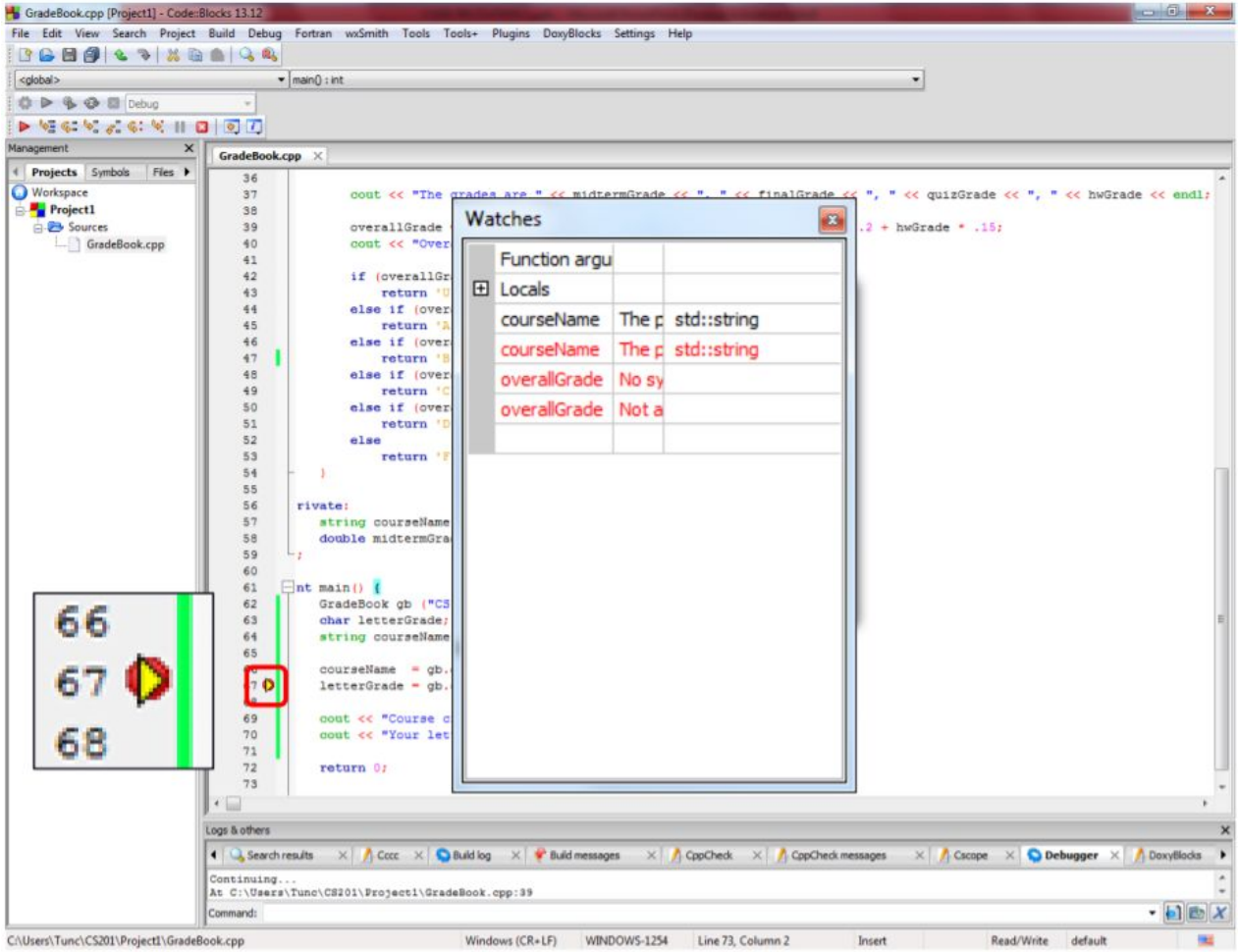
7. Debugging


GradeBook.cpp [Project1] - Code::Blocks 13.12
Watches
Function argu
Locals
courseName
0x22fec0
courseName
<optimized out>,
overallGrade
overallGrade
76
Program received signal SIGSEGV, Segmentation fault.

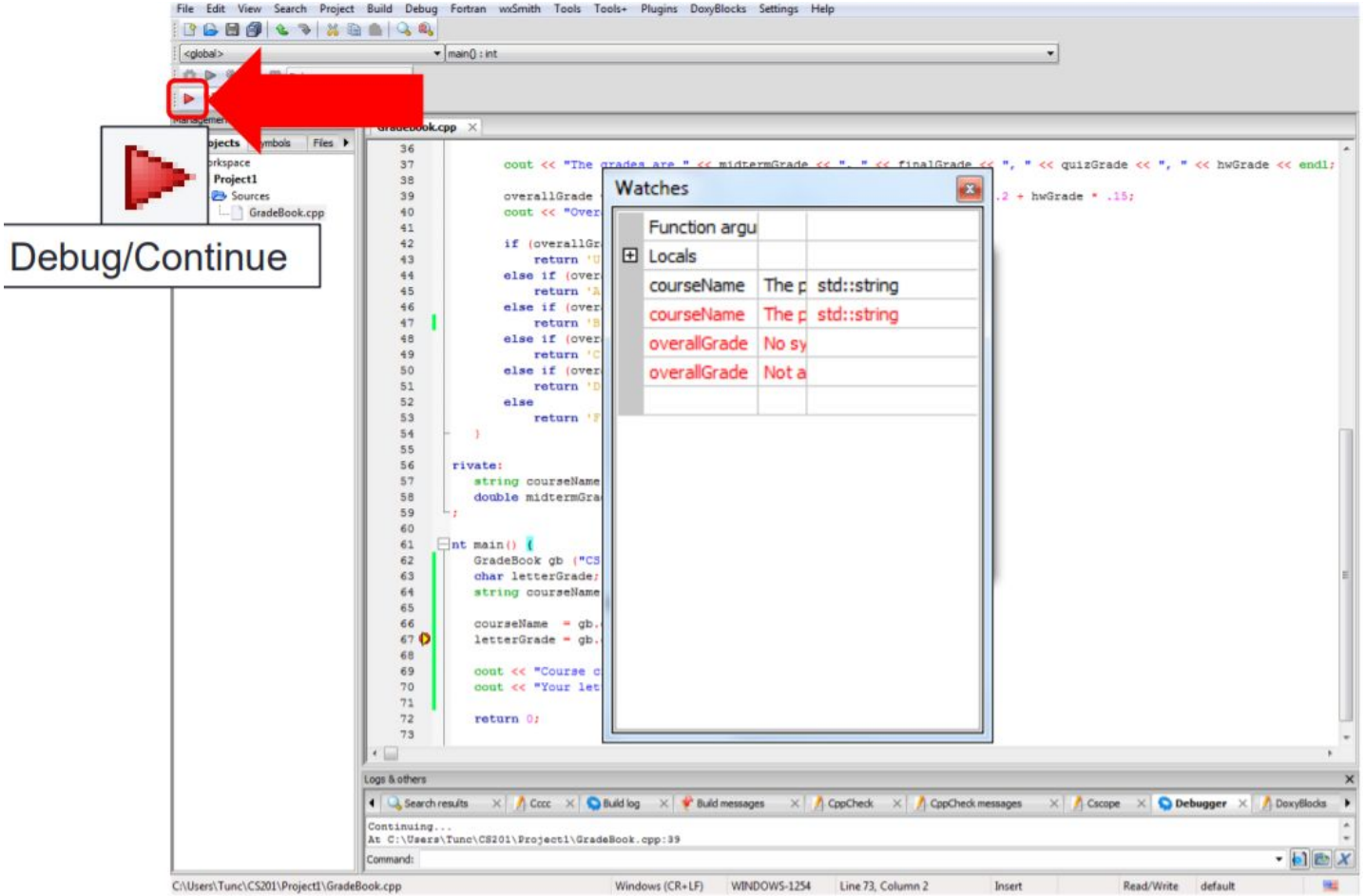
7. Debugging


GradeBook.cpp [Project1] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Step out
Watches
Function argu
Locals
courseName 0x22fec0
courseName <optimized out>,
overallGrade
overallGrade 76 double
Logs & others
Program received signal SIGSEGV, Segmentation fault.
Command:
Continue execution until the current frame exits
Windows (CR+LF) WINDOWS-1254 Line 51, Column 24 Insert Read/Write default

7. Debugging


GradeBook.cpp [Project1] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
Project1
Sources
GradeBook.cpp
Watches
Function argu
Locals
courseName The p std::string
courseName The p std::string
overallGrade No sy
overallGrade Not a
66
67
68
Logs & others
Search results Cccc Build log Build messages CppCheck CppCheck messages Cscope Debugger DoxyBlocks
Continuing...
At C:\Users\Tunc\CS201\Project1\GradeBook.cpp:39
Command:
C:\Users\Tunc\CS201\Project1\GradeBook.cpp
Windows (CR+LF) WINDOWS-1254 Line 73, Column 2 Insert Read/Write default

7. Debugging


Debug/Continue
Watches
Function argu
Locals
courseName
courseName
overallGrade
overallGrade
Logs & others
Debugger
Continuing...
At C:\Users\Tunc\CS201\Project1\GradeBook.cpp:39
Command:

7. Debugging

- Step Into :
  - Runs the program until the next instruction is reached.
- Next Line :
  - Runs the program until the next line of code is reached.
- Step Out :
  - Runs the program until the current procedure is completed.

Step Out ≥ Next Line ≥ Step Into

7. Debugging

At Break Point

```
Output :
Instruction A1
```

```
#include <iostream>
using namespace std;
void procedureB(){
    cout << "Instruction B1" << endl;
    cout << "Instruction B2" << endl;
}
void procedureA(){
    cout << "Instruction A1" << endl;
    procedureB();
    cout << "Instruction A2" << endl;
    cout << "Instruction A3" << endl;
}
int main(){
    cout << "Output : " << endl;
    procedureA();
    cout << "End of Procedure A" << endl;
}
```

Step Into

```
Output :
Instruction A1
```

```
void procedureB(){
    cout << "Instruction B1" << endl;
    cout << "Instruction B2" << endl;
}
```

Next Line

```
Output :
Instruction A1
Instruction B1
Instruction B2
```

```
void procedureA(){
    cout << "Instruction A1" << endl;
    procedureB();
    cout << "Instruction A2" << endl;
    cout << "Instruction A3" << endl;
}
```

Step Out

```
Output :
Instruction A1
Instruction B1
Instruction B2
Instruction A2
Instruction A3
```

```
int main(){
    cout << "Output : " << endl;
    procedureA();
    cout << "End of Procedure A" << endl;
}
```