Human-Centered
Computer Systems Lab

C++ Basics

Lab 9

TA : Changmin Jeon, Minji Kim, Hyunwoo Jung,
Hyunseok Oh, Jingyu Lee, Seungwoo Jo

SEOUL NATIONAL UNIVERSITY

Goal of this Lab

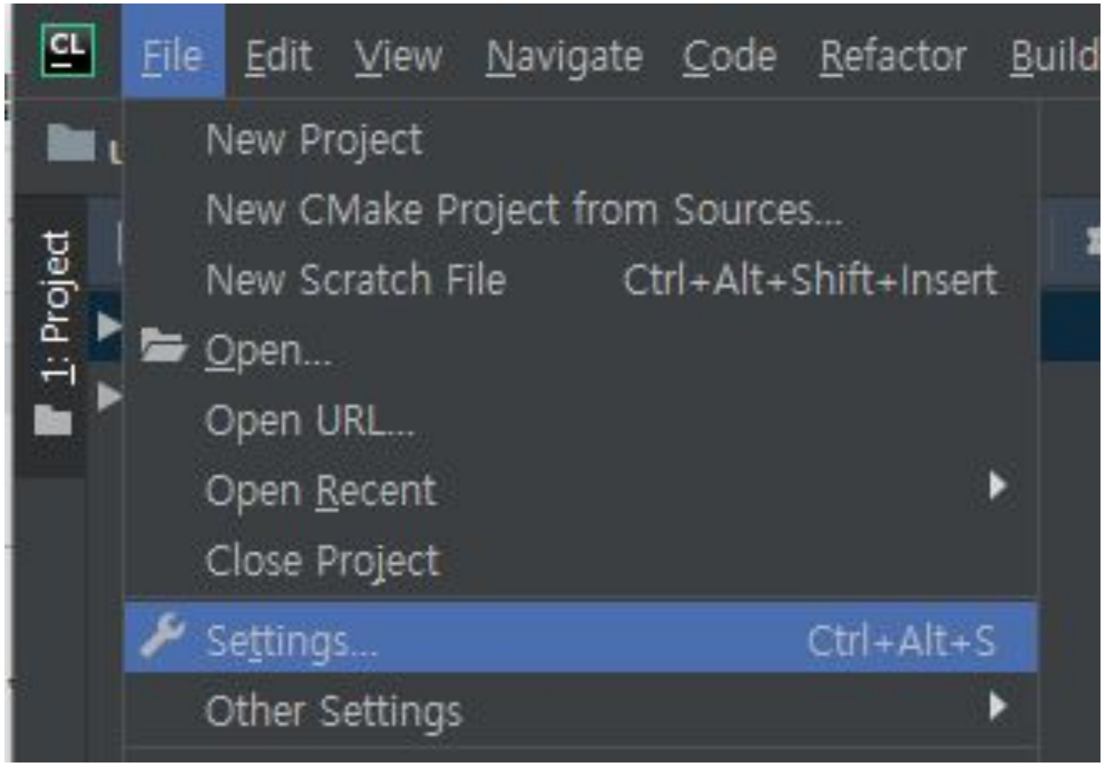
- Understand how to compile C++ program with multiple source files.
- Understand the formatted printing in C++.
- Overview the basic C++ syntax.

Contents

- **Build the program with multiple source files**
- Exercise the formatted print (printf) of the C++

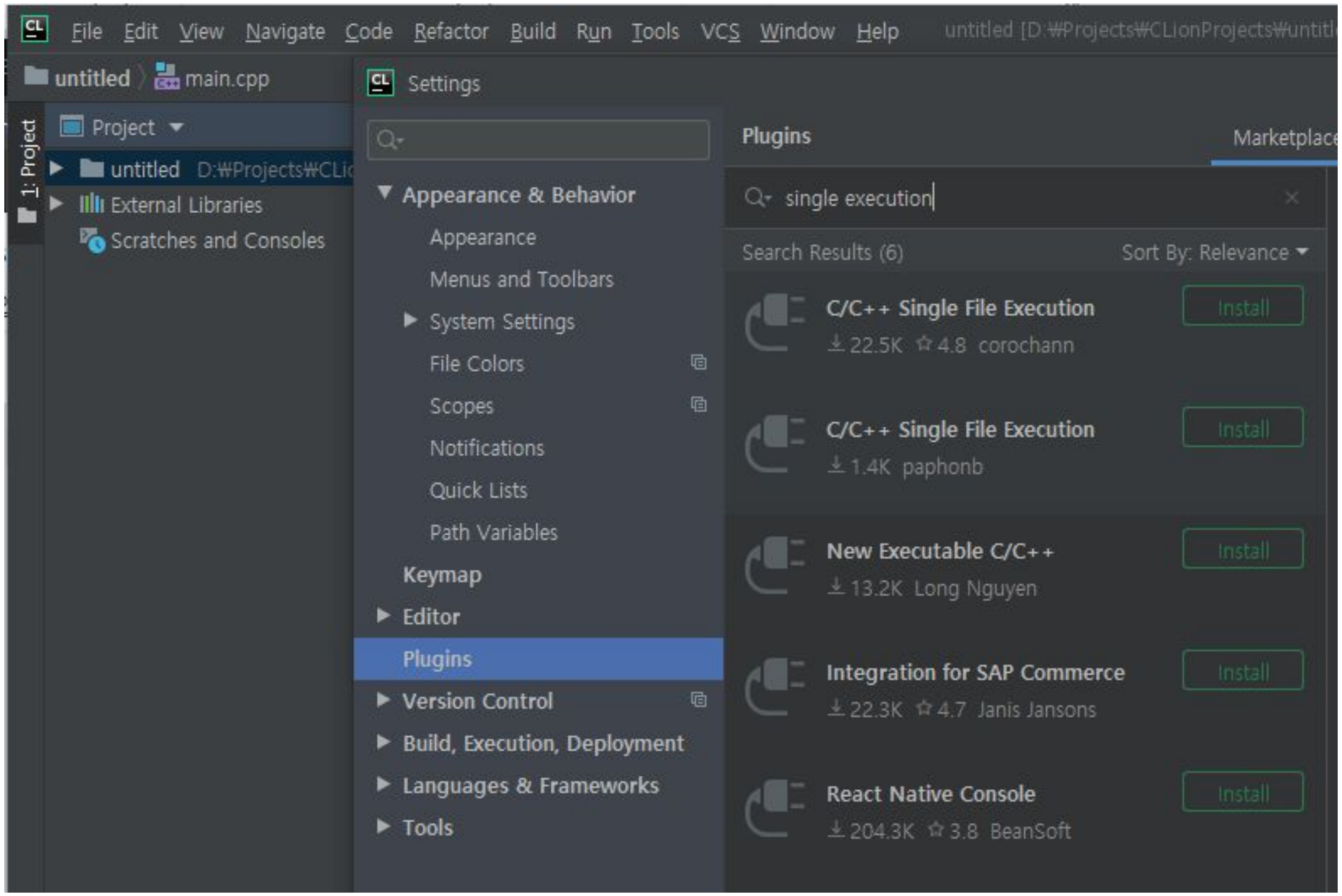
Building multiple single-source files

- File -> Settings



Building multiple single-source files

- Plugins -> search “single execution”



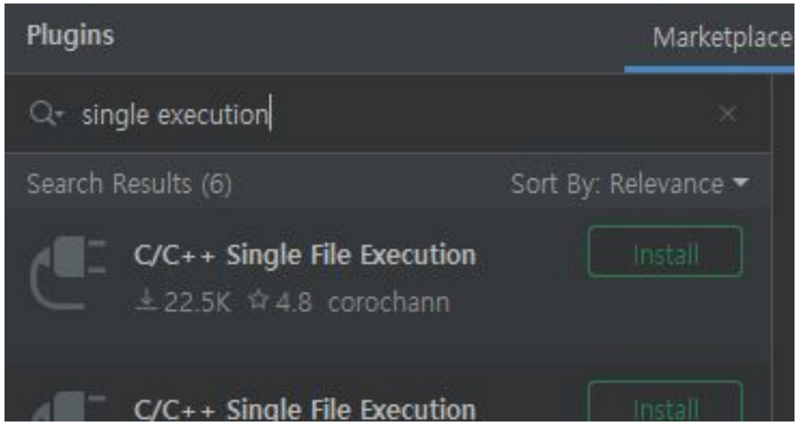
single execution

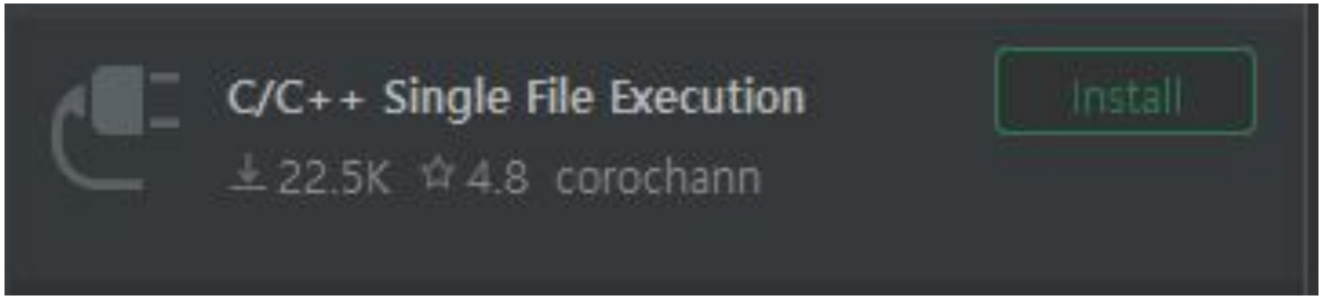
Search Results (6) Sort By: Relevance

Plugin	Downloads	Rating	Author	
C/C++ Single File Execution	22.5K	4.8	corochann	Install
C/C++ Single File Execution	1.4K		paphonb	Install
New Executable C/C++	13.2K		Long Nguyen	Install
Integration for SAP Commerce	22.3K	4.7	Janis Jansons	Install
React Native Console	204.3K	3.8	BeanSoft	Install

Building multiple single-source files

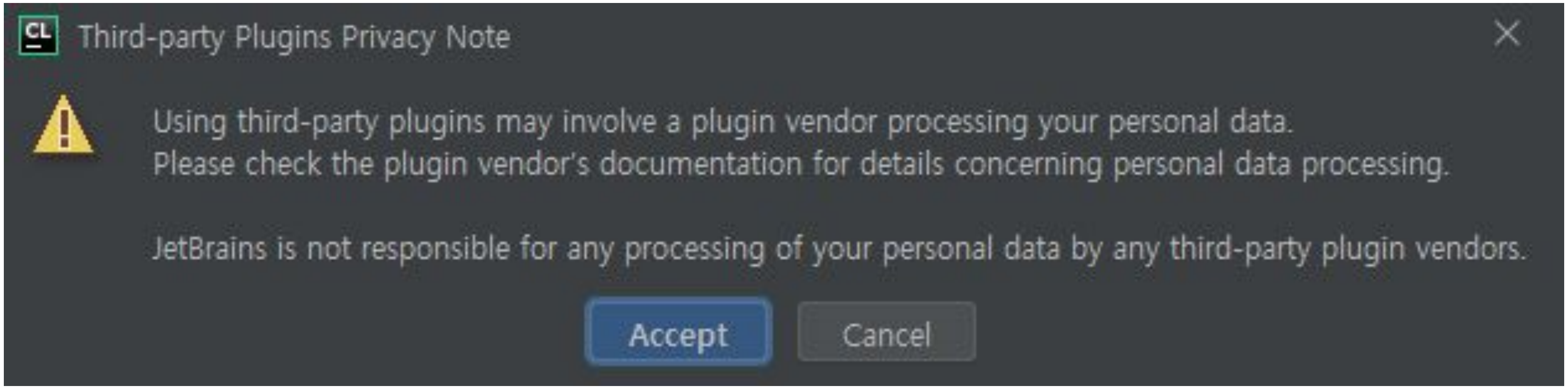
- Install “C/C++ Single File Execution”
    a. Select one with the tag “cocochann”





Building multiple single-source files

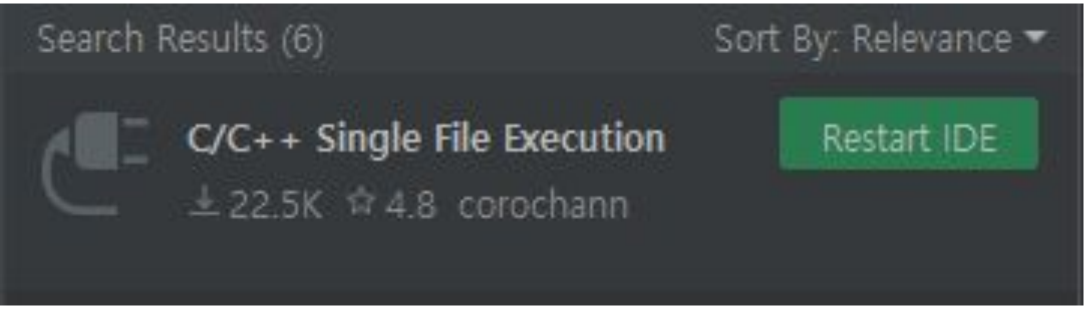
- Accept the third-party plugin privacy Note
- Click "Restart IDE"

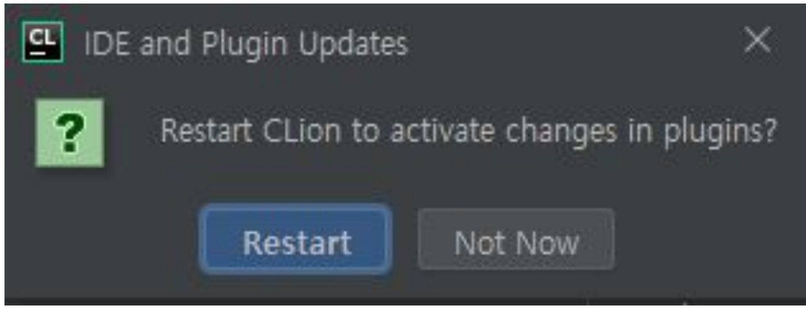


Third-party Plugins Privacy Note

Using third-party plugins may involve a plugin vendor processing your personal data.
Please check the plugin vendor's documentation for details concerning personal data processing.

JetBrains is not responsible for any processing of your personal data by any third-party plugin vendors.



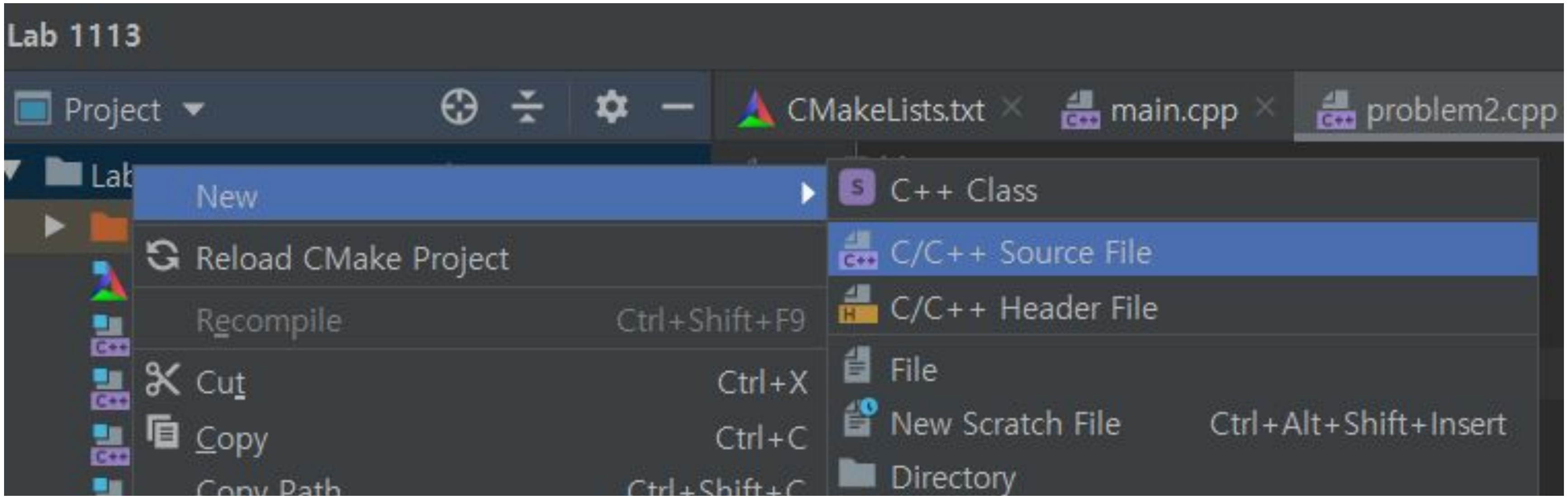


IDE and Plugin Updates

Restart CLion to activate changes in plugins?

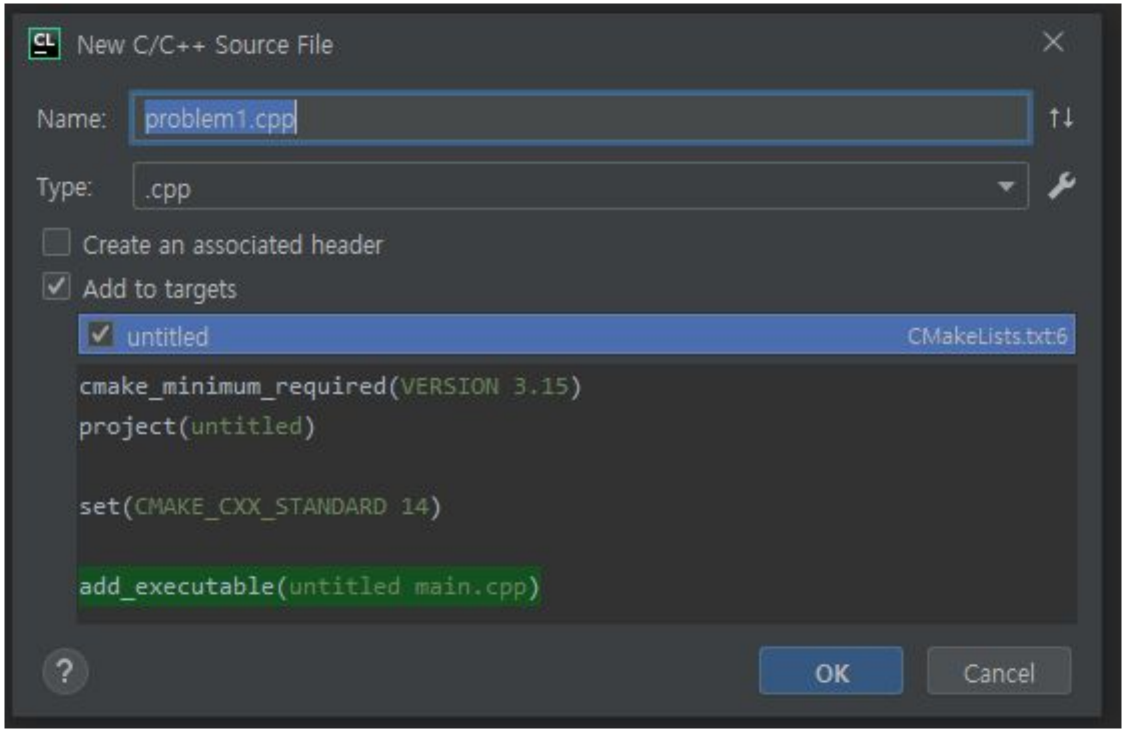
Building multiple single-source files

- Make a new source file



Building multiple single-source files

- Make a new source file



Name: problem1.cpp

Type: .cpp

☐ Create an associated header

☑ Add to targets

☑ untitled CMakeLists.txt:6

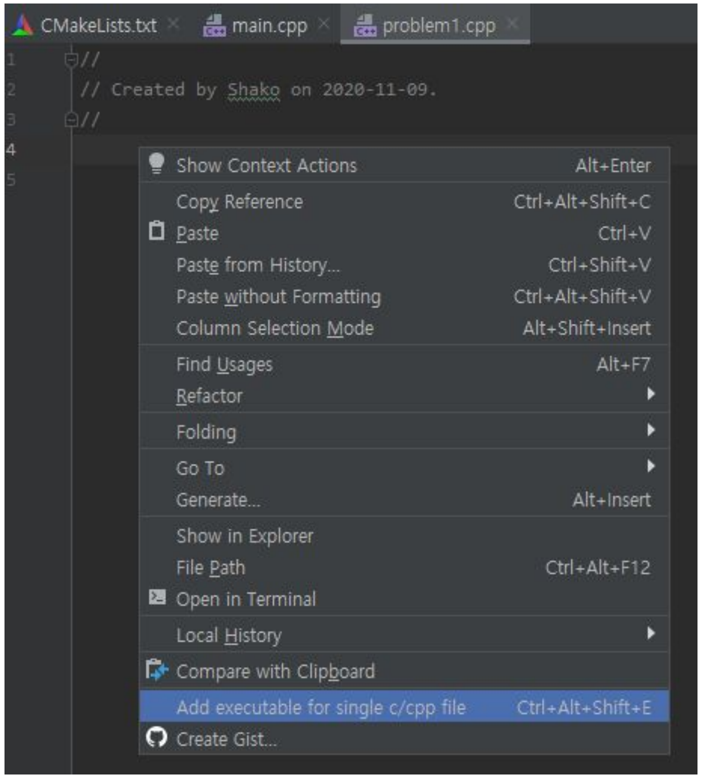
```
cmake_minimum_required(VERSION 3.15)
project(untitled)

set(CMAKE_CXX_STANDARD 14)

add_executable(untitled main.cpp)
```

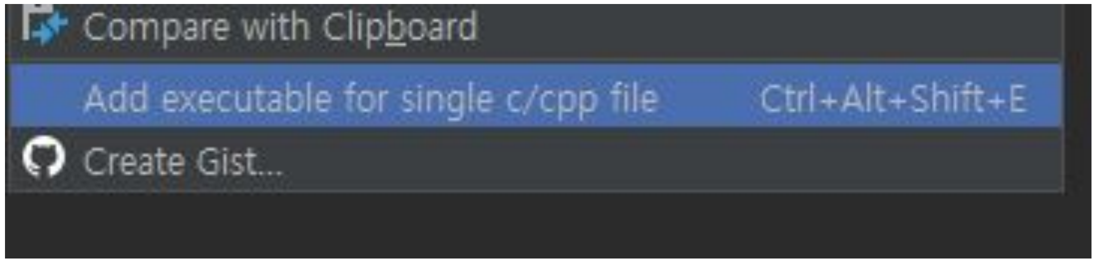
Building multiple single-source files

- Select a file to compile.
- Right click on the editor panel.
- Click “Add executable for single c/cpp file”



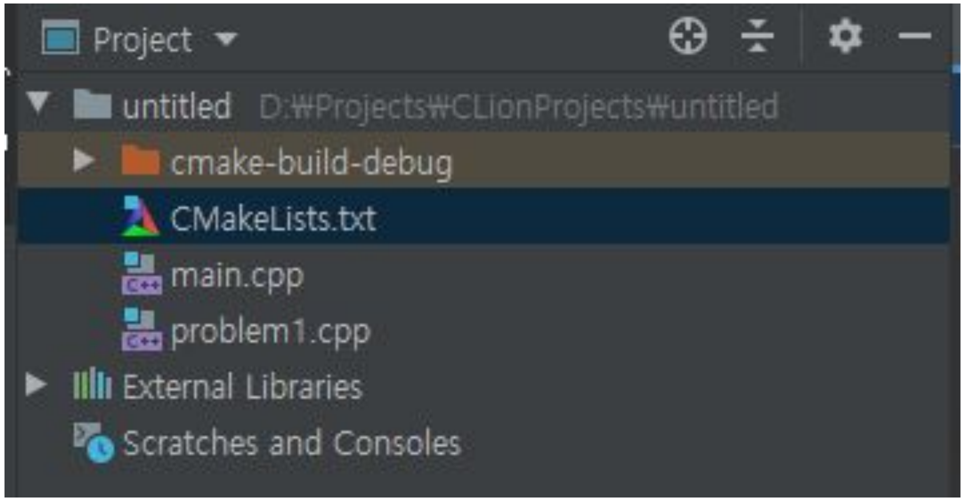
```
//
// Created by Shako on 2020-11-09.
//
```

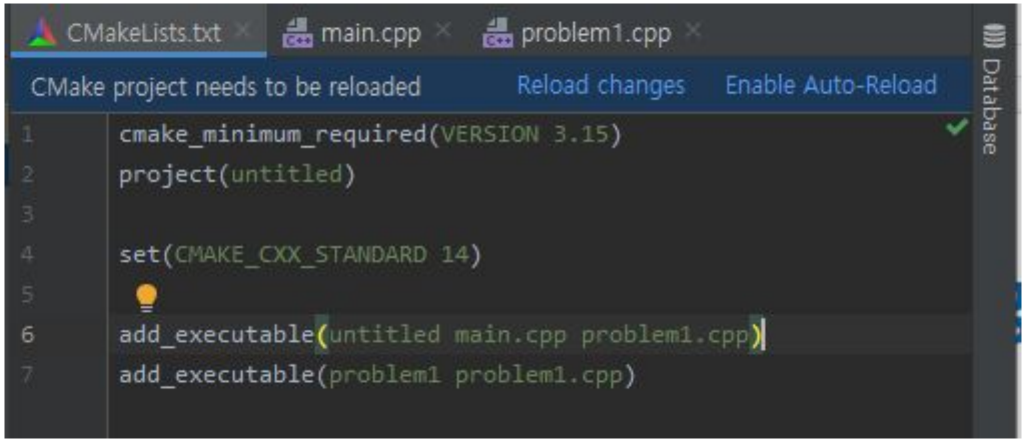
CMakeLists.txt
main.cpp
problem1.cpp
Show Context Actions Alt+Enter
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Paste from History... Ctrl+Shift+V
Paste without Formatting Ctrl+Alt+Shift+V
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Go To
Generate... Alt+Insert
Show in Explorer
File Path Ctrl+Alt+F12
Open in Terminal
Local History
Compare with Clipboard
Add executable for single c/cpp file Ctrl+Alt+Shift+E
Create Gist...



Building multiple single-source files

- Go to CMakeLists.txt and Click Enable Auto-reload





CMake project needs to be reloaded

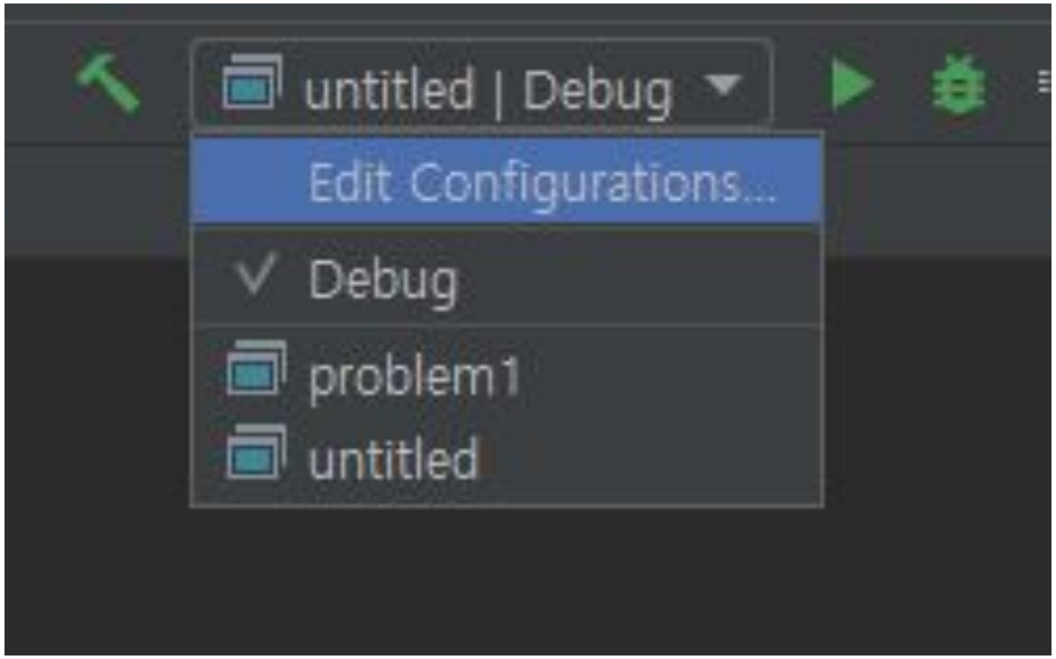
```
cmake_minimum_required(VERSION 3.15)
project(untitled)

set(CMAKE_CXX_STANDARD 14)

add_executable(untitled main.cpp problem1.cpp)
add_executable(problem1 problem1.cpp)
```

Building multiple single-source files

- Choose the main execution target you want.



Contents

- Build the program with multiple source files
- **Exercise the formatted print (printf) of the C++**

cout & endl

- C++ stream object defined to access the standard output is cout.
- cout is used together with insertion operator <<
- endl manipulator can be used to break lines.

```
#include <iostream>
… (in main function)
std::cout << "Output sentence"; // prints Output sentence on screen
std::cout << 120;               // prints number 120 on screen
int x = 99; std::cout << x;               // prints the value of x on screen
std::cout << "This " << " is a " << "single";     // This is a single
std::cout << "First sentence." << std::endl;     // First sentence.
std::cout << "Second sentence." << std::endl;      // Second sentence.
```

cin

- C++ stream object defined to access the standard input is cin.
- cin is used together with extraction operator <<
- cin uses the type of the variable after the << operator to determine the interpretation of input.

```
#include <iostream>
… (in main function)
 int age; float beta;
 std::cin >> age;                          // user input the int value to age
 std::cin >> age >> beta;               // user input the float value to beta
 std::cout << beta << age << std::endl;// print the beta and age value
```

Problem 1. cout & cin

- Objective: Implement the program that doubles the user input integer in problem1.cpp.
- Program Description
  - When the user types in an integer (1 < N < 1000), the integer with double the value should be printed.
  - e.g. input-output pair

Input	1	17	260
Output	2	34	520

printf

- Writes the C string pointed by format to the standard output
- If format includes format specifiers (subsequences beginning with %), the additional arguments are formatted and inserted.
  - New line : \n

```
#include <cstdio>
...(in main function)

printf ("%s \n", "A string");      // A String
printf("Second string \n");              // Second string
printf ("Decimals: %d\n", 1977);         // Decimals : 1977
```

Format String

Additional Arguments

Problem 2. printing with printf

- Objective: Implement the program that prints the given sentence without importing <iostream>, in problem2.cpp
- Program Description
  - Print the sentence “My name is Lincoln.” with a newline (\n) at the end.
  - Do not import <iostream> (#define <iostream>).

Output

```
My name is Lincoln.

```

Format specifiers in printf

- Ascii Encoding : %c
  - 7-bit character code where every single bit represents a unique character

```
printf ("%c %c \n", ‘f’, 72);     // Ascii character
printf ("%d %d \n", ‘g’, 73);    // decimal number of character
```

- decimal, hexadecimal, octal
  - decimal : %d, octal : %o, hexadecimal : %x
  - hex & oct with prefix (0x) : insert # right after %

```
printf ("%d %o %x \n", 1977, 1977, 1977);     // decimal, oct, hex
printf("%#o %#x \n", 1977, 1977);            // hex / oct with prefix 0x
```

Format specifiers in printf

- Minimum width
  - Minimum number of characters to be printed.
  - Padded with blank spaces if printed number < width

```
printf ("%10d%5d%2d\n", 1977, 1977,1977);
//      1977 19771977
```

10 chars 5 4

- Preceding zeros
  - Similar to minimum width, but pad with zeros

```
printf ("%010d%05d%02d\n", 1977, 1977,1977);
//0000001977019771977
```

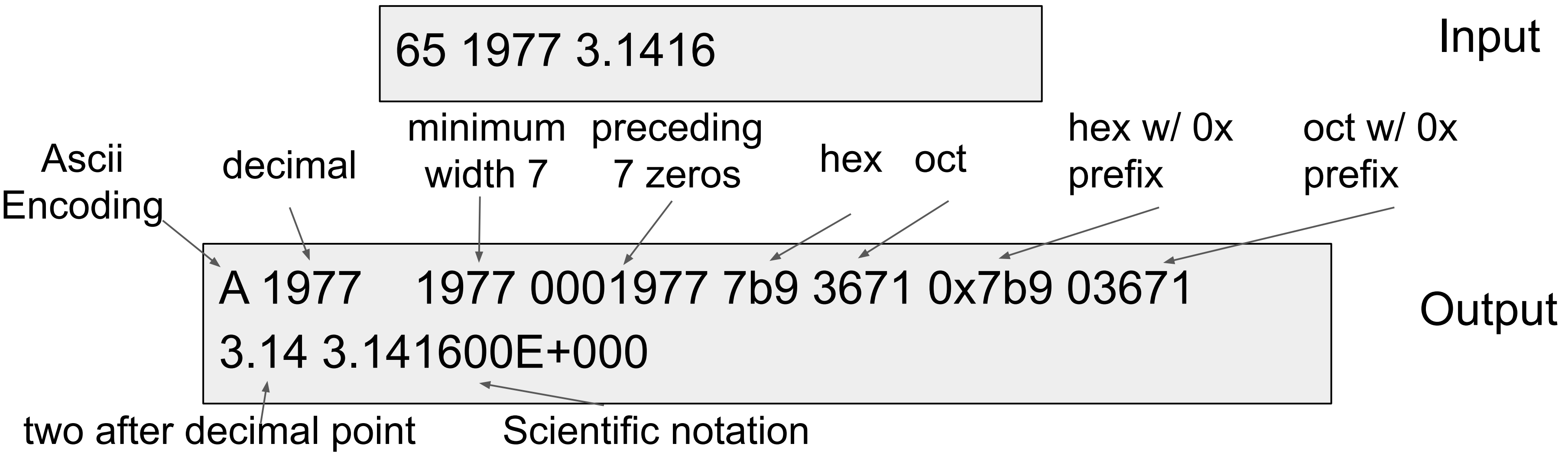
Format specifiers in printf

- Floating point
  - Decimal floating point : %f
  - %N.Mf : minimum width N, decimal precision up to M
    - e.g. %4.2f , %12.6f
  - %E : scientific notation of floats

```
printf ("%f %.3f %E \n", 3.14, 3.14,3.14);
//3.140000 3.140 3.140000E+000
```

Problem 3. Formatted print with printf

- Objective: Implement the program that prints as following format, in problem3.cpp
- Program Description
  - User inputs two integers A, B and a float C (65 <= A <= 90), (0 < B < 3000) , (0.0 <= C <= 1000.0).
  - It prints as following.



Input

```
65 1977 3.1416
```

Output

```
A 1977    1977 0001977 7b9 3671 0x7b9 03671
3.14 3.141600E+000
```

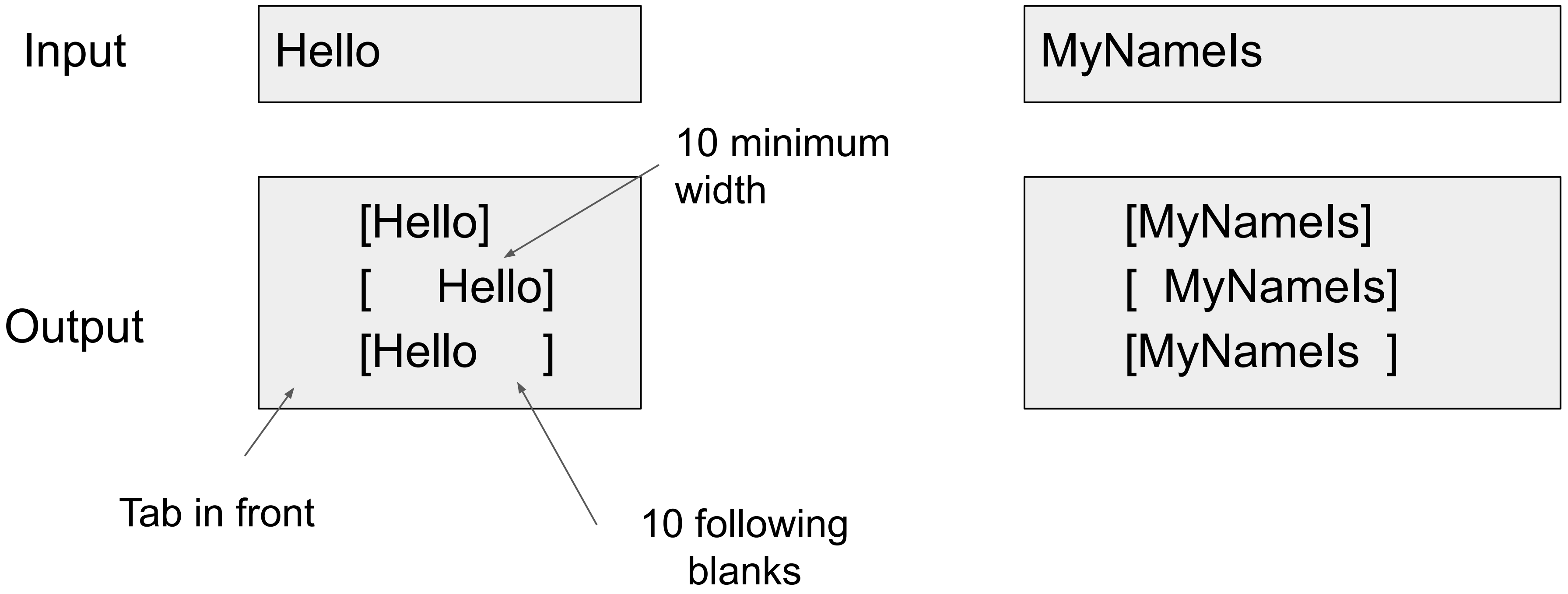
printf on Strings

- %s to print String of characters
  - String literal and char array, but not std::string.
- Minimum width is also applicable
  - - for left alignment (default is right alignment)

```
char str[100] = “Hello there.”;
printf ("%s %s \n", "A string", str);      // A string Hello there
printf("1 %10s 1 \n", “Mine”);             // 1       Mine 1
printf ("1 %-10s 1", “Mine”);              // 1 Mine       1
```

Problem 4 : String print with printf

- Objective: Implement the program that prints input string as following format, in problem4.cpp
- Program Description
  - User inputs a string S.



Input

```
Hello
```

```
MyNameIs
```

Output

```
	[Hello]
	[     Hello]
	[Hello     ]
```

```
	[MyNameIs]
	[  MyNameIs]
	[MyNameIs  ]
```

scanf

- Reads data from standard input.
- Stores them by the format to the additional argument variables.
  - preceding & required for integer and float

```
#include <cstdio>
...(in main function)

 char str [80]; int i;
 printf ("Enter your family name: ");
 scanf ("%79s",str);                    // Stores user input string to str
   Format String   Additional Arguments
 printf ("Enter your age: ");
 scanf ("%d",&i);                       // Stores user input int to i
 printf ("Mr. %s , %d years old.\n",str,i);
```

Problem 5. scanf

- Objective: Implement the program that repeats the input sentence twice without importing <iostream>, in problem5.cpp
- Program Description
    - User inputs a string.
    - The program outputs the string twice in a row in the next line.
    - the <iostream> should not be imported.

Input	Hello	MyNameIs
Output	HelloHello	MyNameIsMyNameIs

Problem 6. Matrix input

- Objective : Implement the program that reads the NxN square matrix elementwise, and pretty print the entire matrix.
- Description :
  - In first line, the dimension integer N is input.
  - In N*N subsequent lines, the prompt “A[i][j]=” is printed and the int value is received.
  - In N subsequent lines, the i-th row integers of the matrix are printed, separated with space.

Example I/O

```
3
A[0][0]=1
A[0][1]=2
A[0][2]=3
A[1][0]=4
A[1][1]=5
A[1][2]=6
A[2][0]=7
A[2][1]=8
A[2][2]=9
1 2 3
4 5 6
7 8 9
```

Submission

- Compress your Project directory into a zip file.
  - It should include problem1.cpp ~ problem5.cpp
- Rename your zip file as 20XX-XXXXX_{name}.zip
  - for example, 2020-12345_KimMinji.zip
- Upload it to eTL - Lab 9 assignment.