CSCI2100C Data Structures

Tutorial 01 Introduction

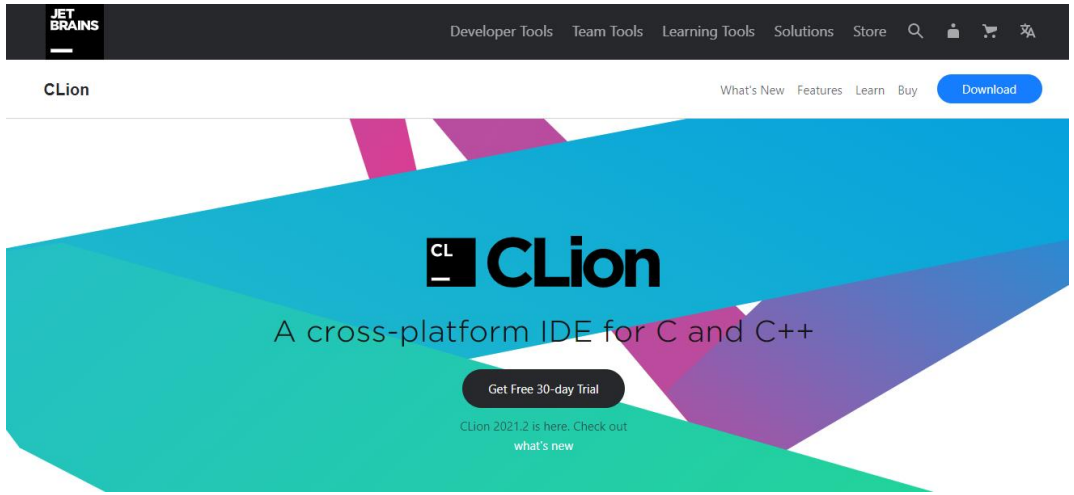
LI Muzhi 李木之
mzli@cse.cuhk.edu.hk

Outline

- Installation and Setup of JetBrains™ CLion
- Installation and Configuration of “gcc” development environment
- Your first C program with CLion.
- The compilation and execution of C program with CLion and command line.

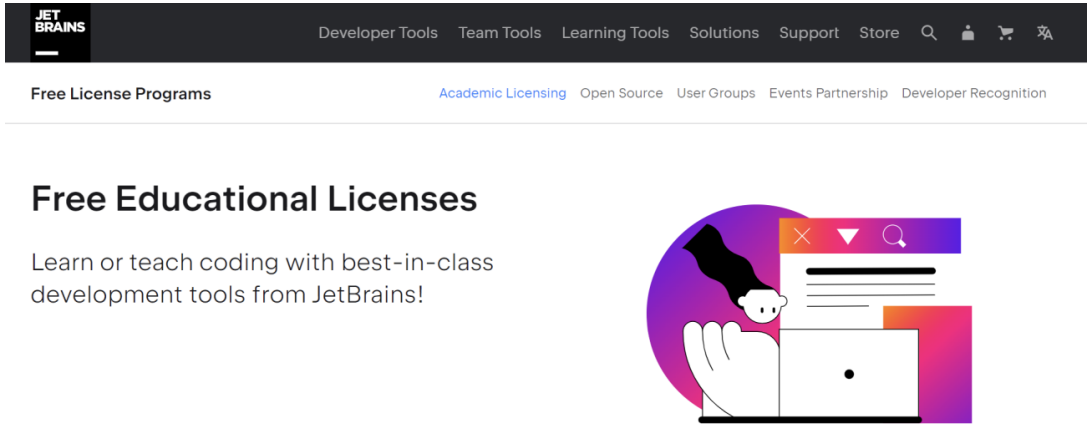
Introduction of JetBrains CLion

- JetBrains™ CLion is a priced software, but university students can apply for a free education account.
- CLion is an IDE for C and C++ which supports Windows, Linux and macOS.



Installation of JetBrains CLion

- Download: https://www.jetbrains.com/clion/download/#section=windows
- For Windows user: Microsoft Visual Studio is required in advance
- For Mac user: Xcode is required in advance.
- Educational License: https://www.jetbrains.com/community/education/#students



Configuration of C Development Environment

- In this course, all your assignments will be compiled by “gcc” compiler.
  - Specific compiler version: gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0, on Windows 10 **or** default gcc on macOS (depends on the TA take in charge).
- For Windows user, you should install *MinGW* (i.e., GNU GCC/G++ compiler and GDB debugger).
  - Website: https://www.mingw-w64.org/downloads/#mingw-builds
- For macOS user, gcc is automatically installed with Xcode.
- For Linux user, you should execute the following command:
  - sudo apt-get install gcc g++ make
- You are free to use any compiler, IDE, and operating system. Please make sure that your code is not platform specific.

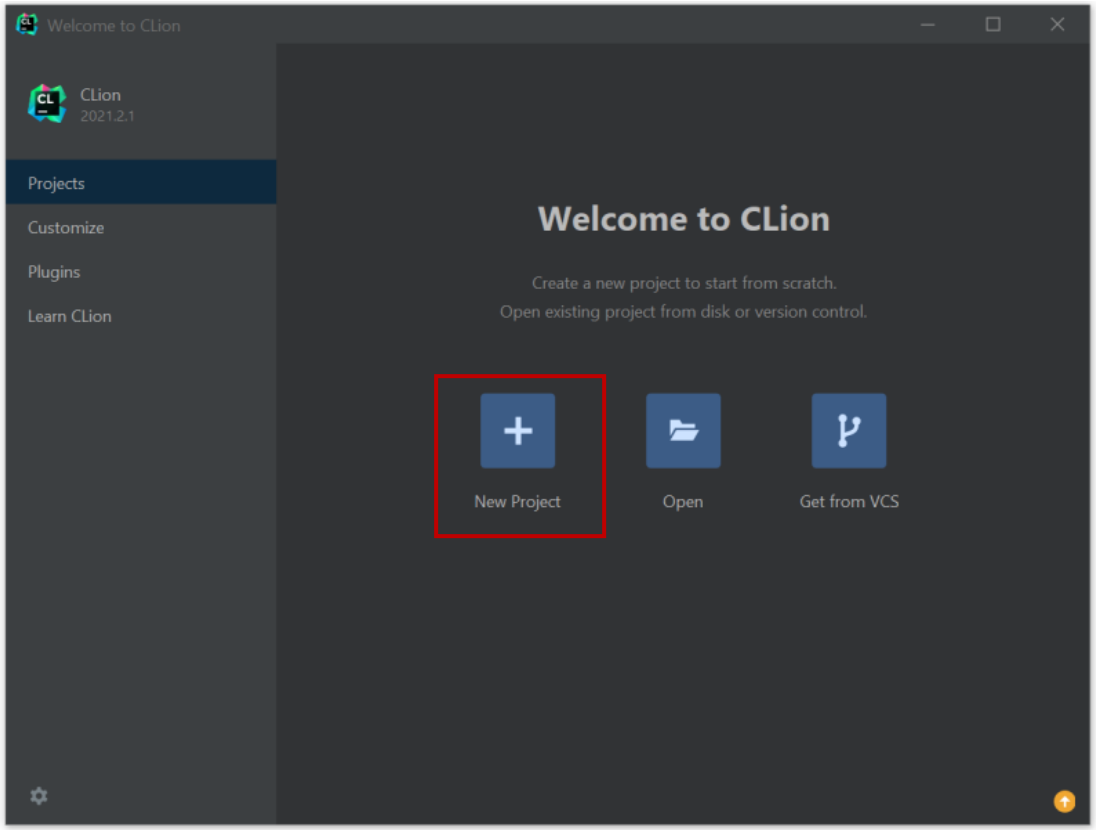
Check whether GCC is successfully installed on your PC

- Use command gcc --version
- I strongly recommend you install g++ at the same time. g++ is a C++ compiler.

```
Microsoft Windows [Version 10.0.19044.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\muzhi>gcc --version
gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


C:\Users\muzhi>g++ --version
g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

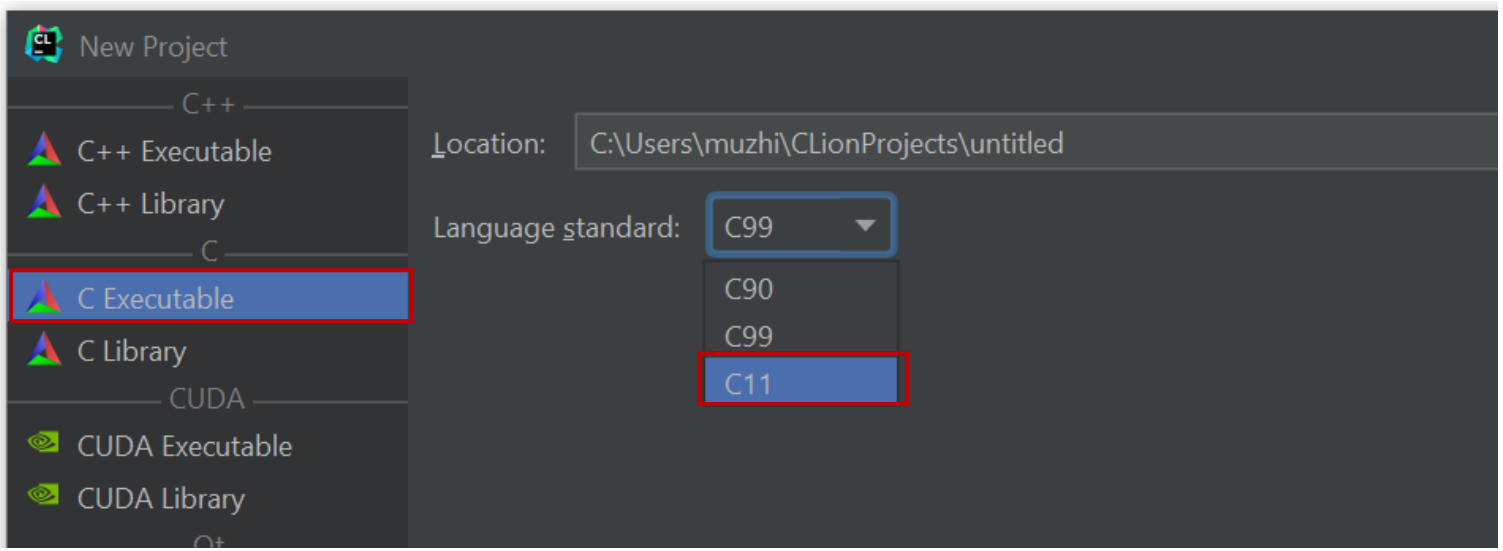
Write Your First Program with CLion

- Press “New Project”.



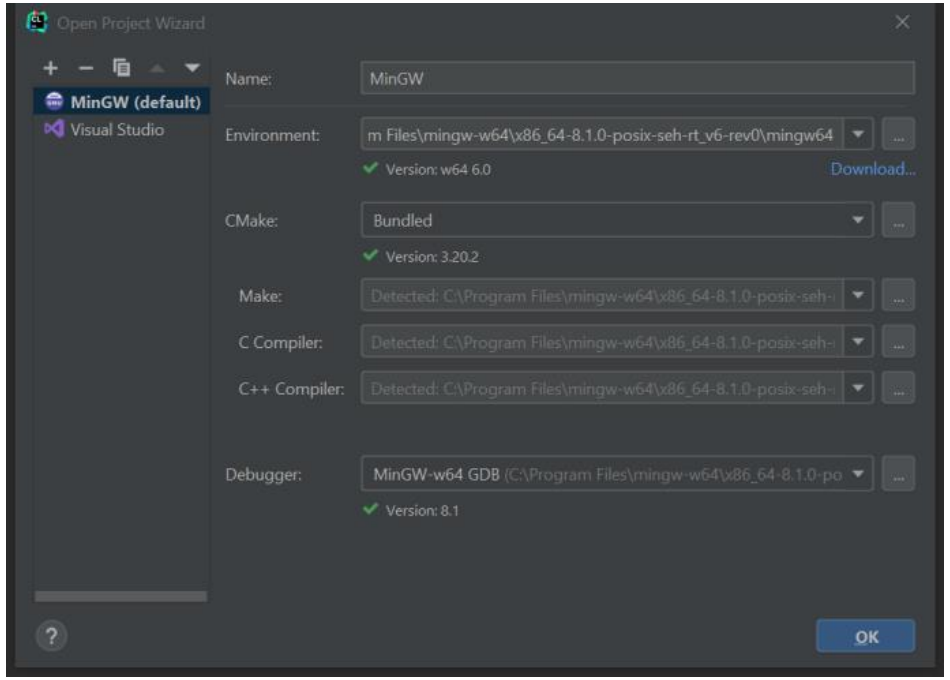
Write Your First C Program with CLion

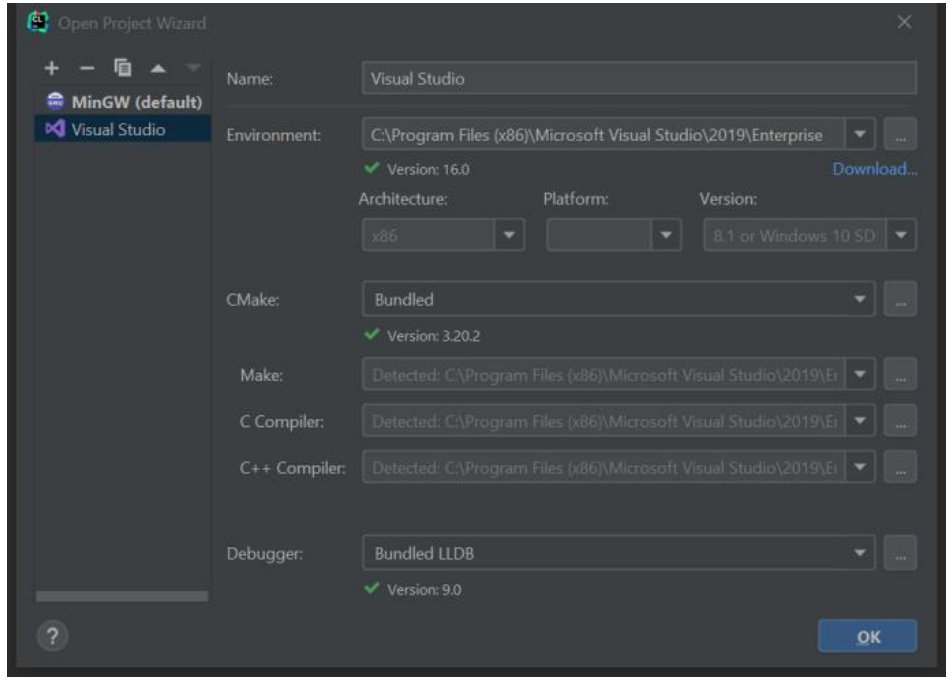
- Select “C Executable” on the left hand side.
- Use “C11” as the compiler standard.



Write Your First C Program with CLion

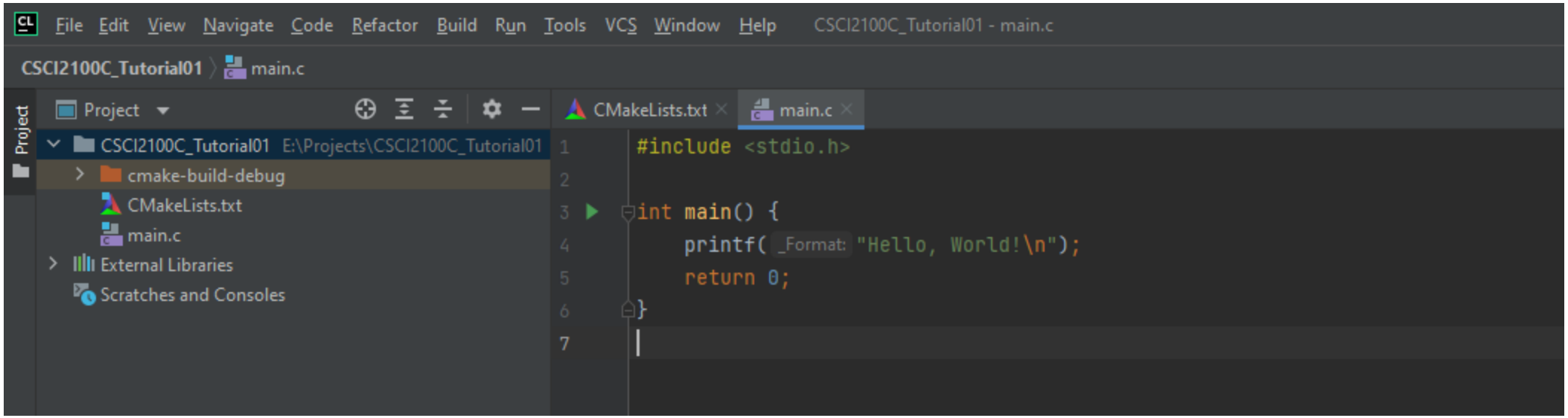
- Normally, if your have correctly installed GCC compiler, CLion will automatically configure the path of CMake, gcc and g++ for you.
- If you want, you can also use other development tools such as MSVC.





Your First C Program with CLion

- In C, the header file for console input and output is <stdio.h>
- The main function should have a integer (int) return type.



Console Input and Output in C

- In C, we use scanf and printf function for console input and console output. (In C, we do not have “cin” and “cout” keyword).
- For example:
  - printf("Hello World!\n"); outputs “Hello World!” to shell with a new line character.
  - scanf function receives user input from the user and save into <u>the memory location</u> of corresponding variable.

```
int a;
scanf("%d", &a);      // scanf("%d", a) is incorrect
```

Cstring

- In C, we do not have a data type called “string”.
- In contrast, we uses character arrays to store strings.
- A `cstring` is an **array of type `char`**, which is terminated by the end-of-string sentinel (or null character) `'\0'` .
  - Anything after `'\0'` is not a part of the string.
- A char-array of length n can store at most n-1 characters.
- For example:

```
char str[20]="Hello World";
```

H	e	l	l	o		W	o	r	l	d	\0								

This character array may store a string with maximum length of 19

Receiving CString Input from Console

- There are two ways to initialize a CString.
- "%s" is the placeholder for string.

```
char str[100];
scanf("%s", str);
//  or using dynamic array
char* str2 = (char *) malloc(100 * sizeof(char));
//  Note: the line of code above is equal to char* str2 = new char[100]; in C++.
scanf("%s", str2);
//  Different from int, there's no '&' before variable name
//  since str or str2 already stores a memory address
free(str2); //  Equals to delete str2; in C++
```

The Following Approach may result in Exception(s).

- Initialize / copy a string to some memory location without space allocation is an *undefined behaviour*.
- The following program does not have any compilation error.
  - Even in some specific case, one may "luckily" find that the program can run without any error.
  - However, the normal operation of the program is not guaranteed…
- Programmers should always avoid "undefined behaviour" in his/er code.

```
char str[5];
scanf("%s", str); // The user inputted a string longer than 4.
//  or using dynamic array
char* str2; // No memory allocation
scanf("%s", str2);
```

Console Input and Output in C

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help CSCI2100C_Tutorial01 - main.c

CSCI2100C_Tutorial01

Project

CSCI2100C_Tutorial01 E:\Projects\CSCI2100C_Tutorial01
- cmake-build-debug
- CMakeLists.txt
- main.c
- External Libraries
- Scratches and Consoles

CMakeLists.txt | main.c

```c
#include <stdio.h>
#include <string.h>

int main() {
    int year;
    char month[4];
    int day;
    double time;
    printf( _Format: "Please input the current date and time (e.g. 12 JAN 2022 12:30):");
    scanf( _Format: "%d %s %d %lf", &day, month, &year, &time);
    printf( _Format: "Now is %d %s %d %.2lf", year, month, day, time);
    return 0;
}
```

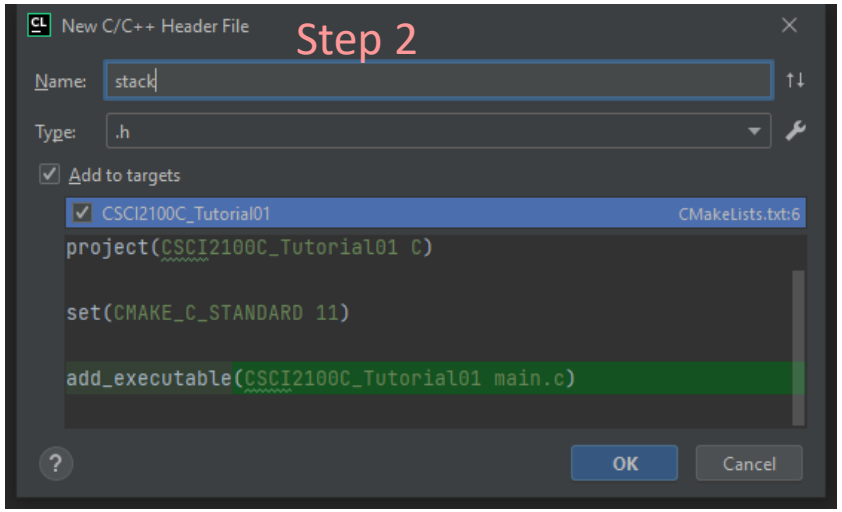
Run: CSCI2100C_Tutorial01

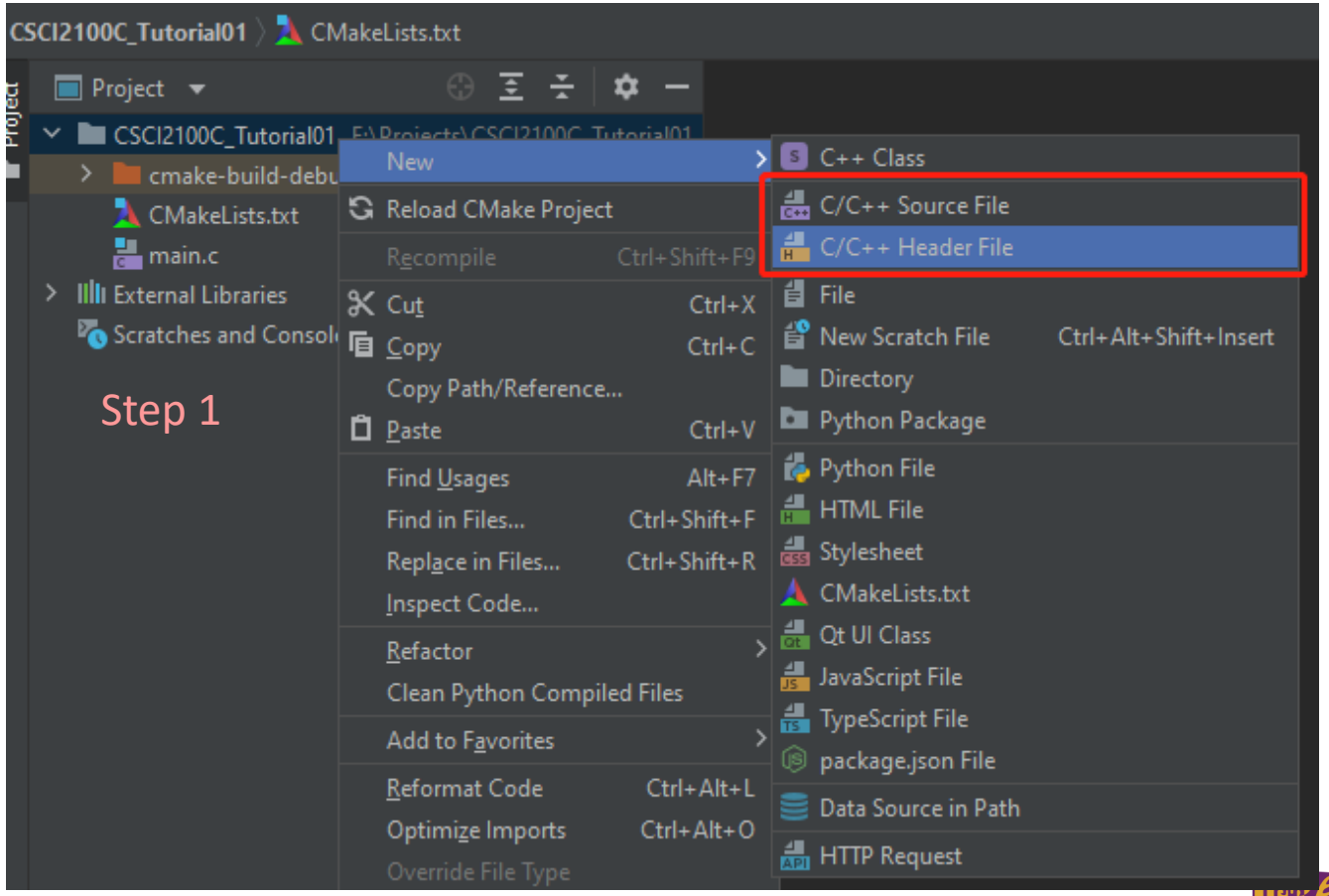
```
E:\Projects\CSCI2100C_Tutorial01\cmake-build-debug\CSCI2100C_Tutorial01.exe
Please input the current date and time (e.g. 12 JAN 2022 12:30):12 JAN 2022 16.30
Now is 2022 JAN 12 16.30
Process finished with exit code 0
```

I strongly suggest you try this example in your IDE unless you are familiar with C programming.

Header Files and Source Code Files

- How to create a new header / source code file in Clion.
- Right Click -> New -> C/C++ Source (Header) File





Example: stack.h

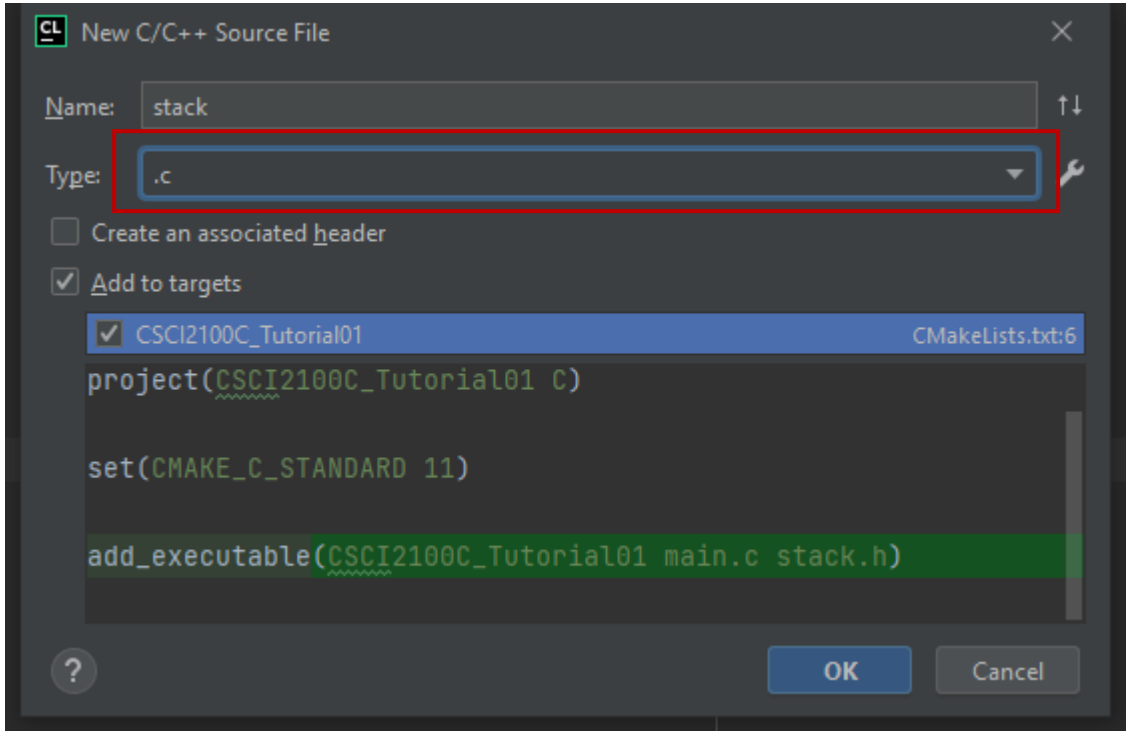
- Function templates should be placed in the header file(s).

stack.h

```
//
// Created by muzhi on 8/1/2022.
//

#ifndef CSCI2100C_TUTORIAL01_STACK_H
#define CSCI2100C_TUTORIAL01_STACK_H

typedef struct stackCDT *stackADT;
typedef int stackElementT;
stackADT EmptyStack(void);
void Push(stackADT stack, stackElementT element);
stackElementT Pop(stackADT stack);
int StackDepth(stackADT stack);
int StackIsEmpty(stackADT stack);

#endif //CSCI2100C_TUTORIAL01_STACK_H
```

Example: stack.c

- Remember to choose ".c" as the file type.



- The implementation of functions should be placed in a separate .c source code file.
- The source code file stack.c should include the header file we have created before.

stack.h stack.c calculator.c

```c
#include <stdlib.h>
#include "stack.h"
struct stackCDT { stackElementT elements[100]; int count;};
stackADT EmptyStack(void) {
    stackADT stack; stack=(stackADT)malloc(sizeof(*stack)); stack->count=0;
    return(stack);
}
void Push(stackADT stack, stackElementT element)
{ stack->elements[(stack->count)++]=element; }
stackElementT Pop(stackADT stack)
{ return(stack->elements[--(stack->count)]); }
int StackDepth(stackADT stack) { return(stack->count); }
int StackIsEmpty(stackADT stack) { return(stack->count==0); }
```

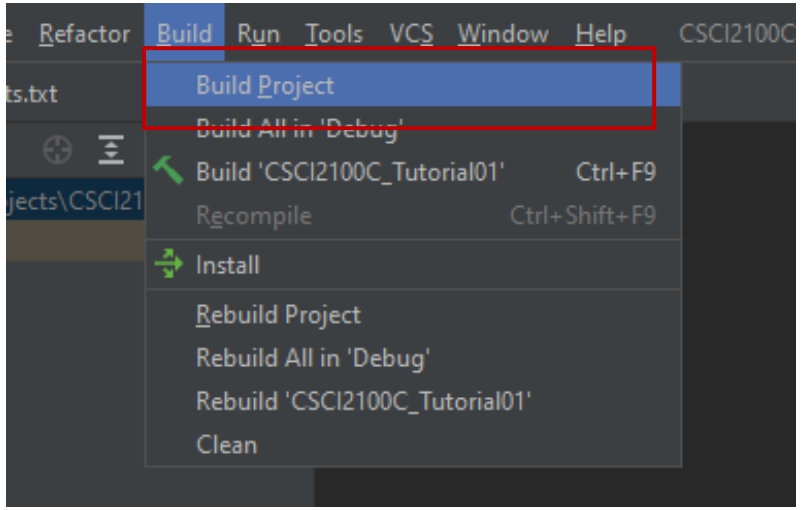
Example: calculator.c

- Let's build a calculator based on the abstract data type stack.
- The calculator.c is not aware of the implementation of data structure stack.

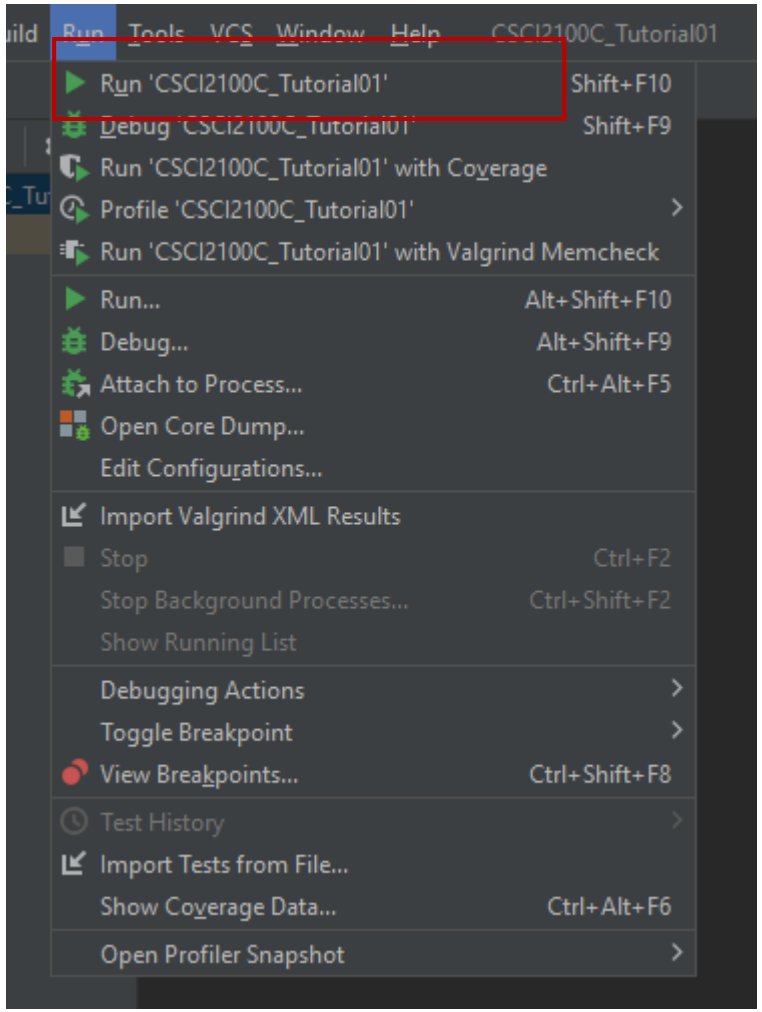
```c
 * File: calculator.c
 */
#include <stdio.h>
#include <stdlib.h>
#include "stack.h"

static void ApplyOperator(char op, stackADT operandStack)
{
    stackElementT lhs, rhs, result;
    rhs = Pop(operandStack);
    lhs = Pop(operandStack);
    switch (op) {
        case '+' : result = lhs + rhs; break;
        case '-' : result = lhs - rhs; break;
        case '*' : result = lhs * rhs; break;
        case '/' : result = lhs / rhs; break;
    }
    Push(operandStack, result);
}
```

```c
int main() {
    stackADT operandStack;
    char op, line[80];
    operandStack = EmptyStack();
    do {
        scanf( _Format: "%s", line);
        op = line[0];
        switch (op) {
            case '+' : case '-' : case '*' : case '/' :
                ApplyOperator(op, operandStack); break;
            case '=' :
                printf( _Format: "%d\n", Pop(operandStack)); break;
            default :
                Push(operandStack, atoi(line));
        }
    } while (op != '=');
    return 0;
}
```

The Compilation and Execution of C program with CLion




```
int main() {
    stackADT operandStack;
    char op, line[80];
    operandStack = EmptyStack();
    do {
        scanf( _Format: "%s", line);
        op = line[0];
```



The Compilation of C program with gcc using Command Line

- Use gcc command: gcc [filename1 filename2 …] -o output.exe

gcc calculator.c stack.c -o calculator.exe

```
E:\Projects\CSCI2100C_Tutorial01>dir
 Volume in drive E is SSD
 Volume Serial Number is [redacted]

 Directory of E:\Projects\CSCI2100C_Tutorial01

08/01/2022  23:23    <DIR>          .
08/01/2022  23:23    <DIR>          ..
08/01/2022  23:23    <DIR>          .idea
08/01/2022  23:18             1,012 calculator.c
08/01/2022  23:20    <DIR>          cmake-build-debug
08/01/2022  23:06               162 CMakeLists.txt
08/01/2022  23:23               533 stack.c
08/01/2022  22:54               410 stack.h
               4 File(s)          2,117 bytes
               4 Dir(s)  21,542,256,640 bytes free

E:\Projects\CSCI2100C_Tutorial01>gcc calculator.c stack.c -o calculator.exe

E:\Projects\CSCI2100C_Tutorial01>
```

If you are using macOS, the extension of executable file will be different.

The Compilation of C program with MSVC using Developer Command Line

- Use cl command: cl /EHsc [filename1 filename2 …]

cl /EHsc calculator.c stack.c

```
E:\Projects\CSCI2100C_Tutorial01>cl /EHsc calculator.c stack.c
Microsoft (R) C/C++ Optimizing Compiler Version 19.29.30138 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

calculator.c
stack.c
Generating Code...
Microsoft (R) Incremental Linker Version 14.29.30138.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:calculator.exe
calculator.obj
stack.obj

E:\Projects\CSCI2100C_Tutorial01>_
```

Note: We will use GCC to compile your assignment solutions.

The Execution of C program with Command Line

- Use command: .\[executable.exe]

Command Prompt - .\calculator.exe

```
E:\Projects\CSCI2100C_Tutorial01>dir
 Volume in drive E is SSD
 Volume Serial Number is [redacted]

 Directory of E:\Projects\CSCI2100C_Tutorial01

08/01/2022  23:24    <DIR>          .
08/01/2022  23:24    <DIR>          ..
08/01/2022  23:23    <DIR>          .idea
08/01/2022  23:18             1,012 calculator.c
08/01/2022  23:24            55,822 calculator.exe
08/01/2022  23:20    <DIR>          cmake-build-debug
08/01/2022  23:06               162 CMakeLists.txt
08/01/2022  23:23               533 stack.c
08/01/2022  22:54               410 stack.h
               5 File(s)         57,939 bytes
               4 Dir(s)  21,542,002,688 bytes free

E:\Projects\CSCI2100C_Tutorial01>.\calculator.exe
```

The executable file generated by the compiler.


The Chinese University of Hong Kong
Q & A