

# CSC 3210

## Computer Organization and Programming

### Lab Work 13

Dr. Zulkar Nine

[mnine@gsu.edu](mailto:mnine@gsu.edu)

Georgia State University

Fall 2021

# Learning Objective

## Procedures

# Disclaimer

- The process shown in these slides might not work in every single computer due to Operating system version, Microsoft Visual Studio versions and everything.
- If you find any unusual error, you can inform the instructor.
- Instructor will help you resolve the issue.

# Attendance!

# Lab Work 13 Instructions

- Lab 13 : Write a search function

# Passing Register Arguments to Procedures:

**Example 1:** Write a procedure that adds the passed parameters (eax,edx, and ecx) and returns the result in eax. The main procedure is shown below. Then execute the program and trace its execution and report exa value.

```
.data
theSum  DWORD  ?
.code
main PROC
    mov  eax,10000h    ; argument
    mov  ebx,20000h    ; argument
    mov  ecx,30000h    ; argument
    call Sumof         ; EAX = (EAX + EBX + ECX)
    mov  theSum,eax    ; save the sum
```

Remember to use **step into (Fn+F11)** when tracing a procurer's call

# Passing Register Arguments to Procedures and Calling External procedures:

**Example 2:** Write the following program

Remember to use **step into** when tracing a procedure's call

```
.386
.model flat,stdcall
.stack 4096
ExitProcess proto,dwExitCode:dword
```

```
.code
```

```
main PROC
```

```
mov ebx,1431    ; pass two parameters, in order
mov ecx,63
```

```
call AddTwo     ; look for return value in EAX
```

```
invoke ExitProcess,0
```

```
main ENDP
```

```
AddTwo PROC
```

```
Mov eax,0
```

```
add eax,ebx
```

```
add eax,ecx
```

```
ret
```

```
AddTwo ENDP
```

```
end main
```

# Lab 13

## Submission



# Submission (1)

```
.data
    array DWORD 10h,20h, 30h, 40h, 50h
    sample DWORD 50h
.code
```

- Write a function that search an item in the array
- The .data section contains two items – “array” and “sample”
  - You have to write a procedure that search “sample” in the “array”.
- If the item is found return the item using EAX register, otherwise, return -1 using EAX register.
- You can pass the parameters using registers
- Push the register arguments at the beginning of the procedure. And pop them at the end of the procedure.
- Detail documentation of the procedure is in the next slide

## Submission (2)

```
;-----  
; This Procedure search a given item in a given array  
; Receives: EBX, ESI, ECX, EDX content as input.  
;   EBX contains the item we want to search on the array  
;   ESI contains the address of the first item in the array  
;   ECX contains the number of items in the array  
;   EDX contains the size of each item in the array  
; Returns : EAX,  
;   If the item found in the array, EAX stores the item we are searching  
;   Otherwise, EAX contains -1  
; Requires: Nothing  
;-----  
Search PROC  
    ; implement the procedure  
    ret  
Search ENDP
```

# Submission(3)

- **Skeleton code for the procedure**

```
Search PROC
```

```
    ; save a copy of the registers except eax in the stack
```

```
    ; Implement Search here
```

```
    ; re-store the items from the stack
```

```
    ret
```

```
Search ENDP
```

# Submission (3)

- Submit the screenshot of your code.
- Debug your code until you reach INVOKE ExitProcess, 0
- Take a screenshot of the watch window showing EAX content
  - Submit the screenshot.
- Also, Rename the asm file using your last name as Lastname.asm
  - Submit the ASM file as well.

# Appendix A

## Checking Memory Data

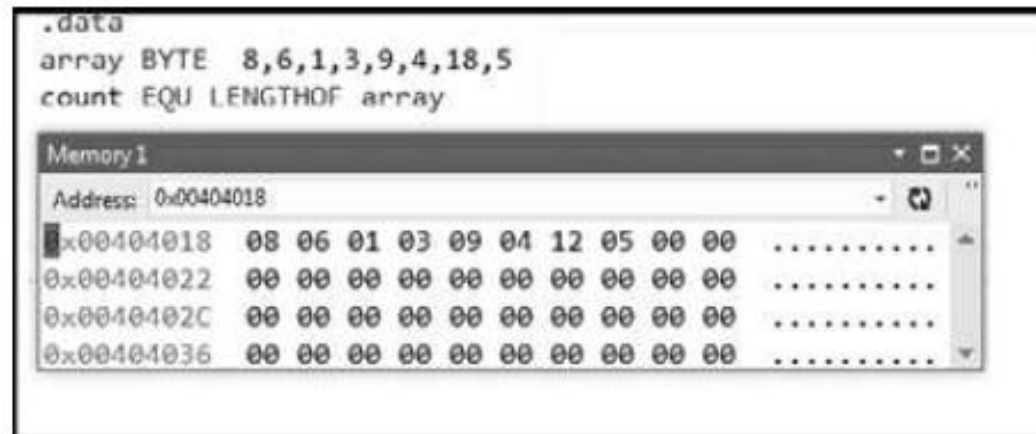
# Checking Memory Data

- Use **Memory window** to verify the values of memory locations.
  - **To activate Memory window**, run the debugger, go to debug menu and click on windows, open it, go to **Memory** then choose **Memory1**.
    - When you run your program and step over every line you will see the changed values marked with red color.

**You Must be in the Debugging Mode to see the memory or the register window**

# Checking Memory Data

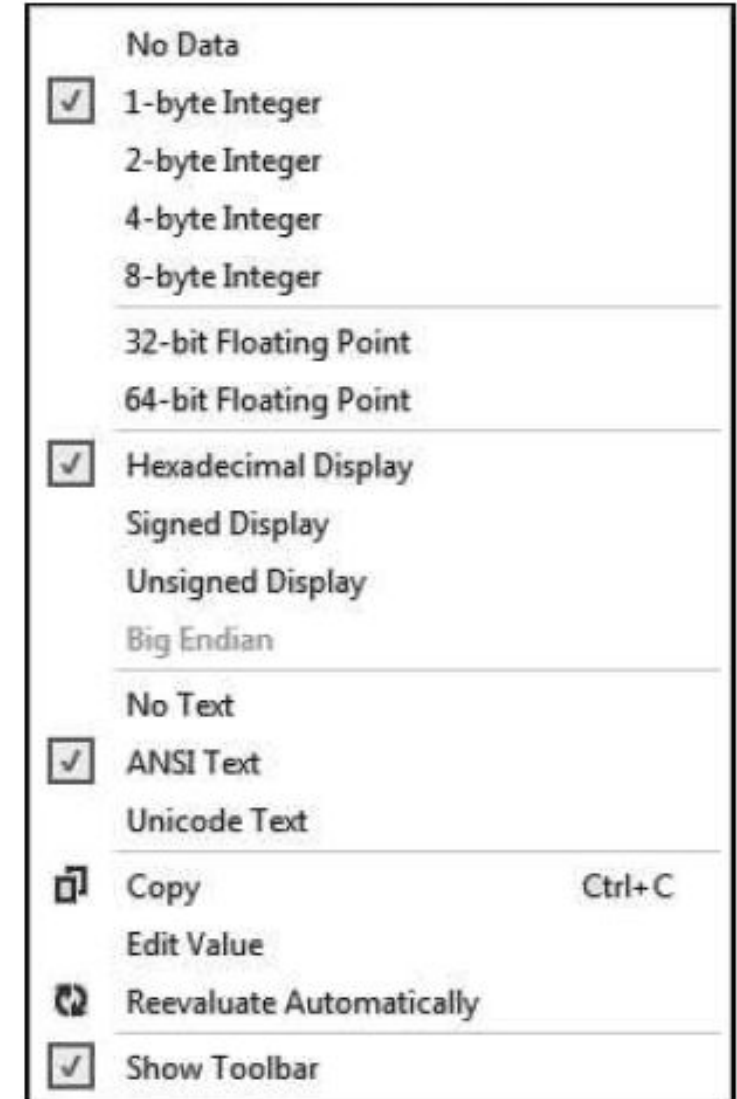
- To activate Memory window,
  - if you want to see the location of your variable in the memory,
    - Memory window search box (on the top of the memory window, Address:)
    - write **&** follow it with the variable name: example: **&myVall**.
    - This will take you to the memory locations of your program (.data section).



# Checking Memory Data

- To activate Memory window,

- You can right-click inside the memory window
- You will see **Popup menu for the debugger's memory window**
- You can choose how you want to group your bytes: by 1,2,4, or by 8
- You can also presents data in **hexadecimal, signed, or unsigned** display



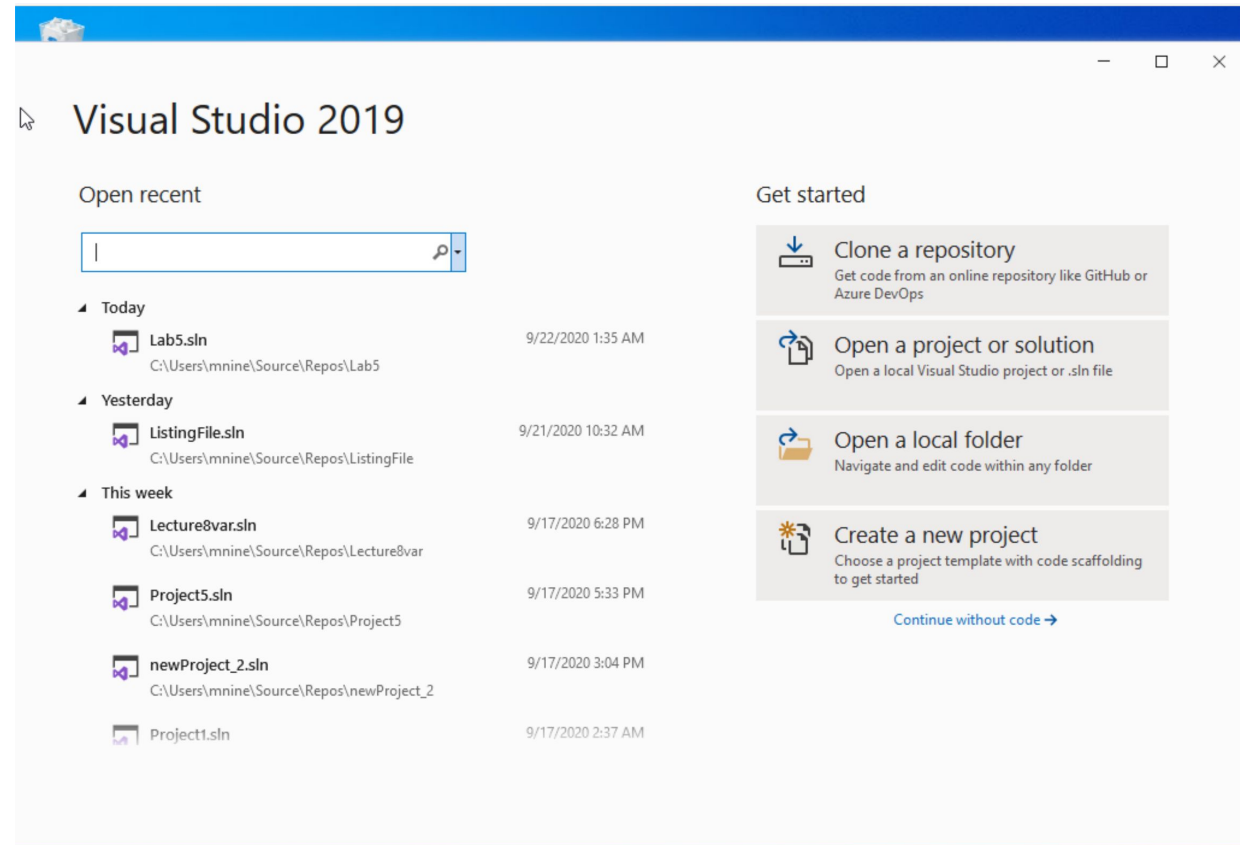


# Appendix B

Create a Project

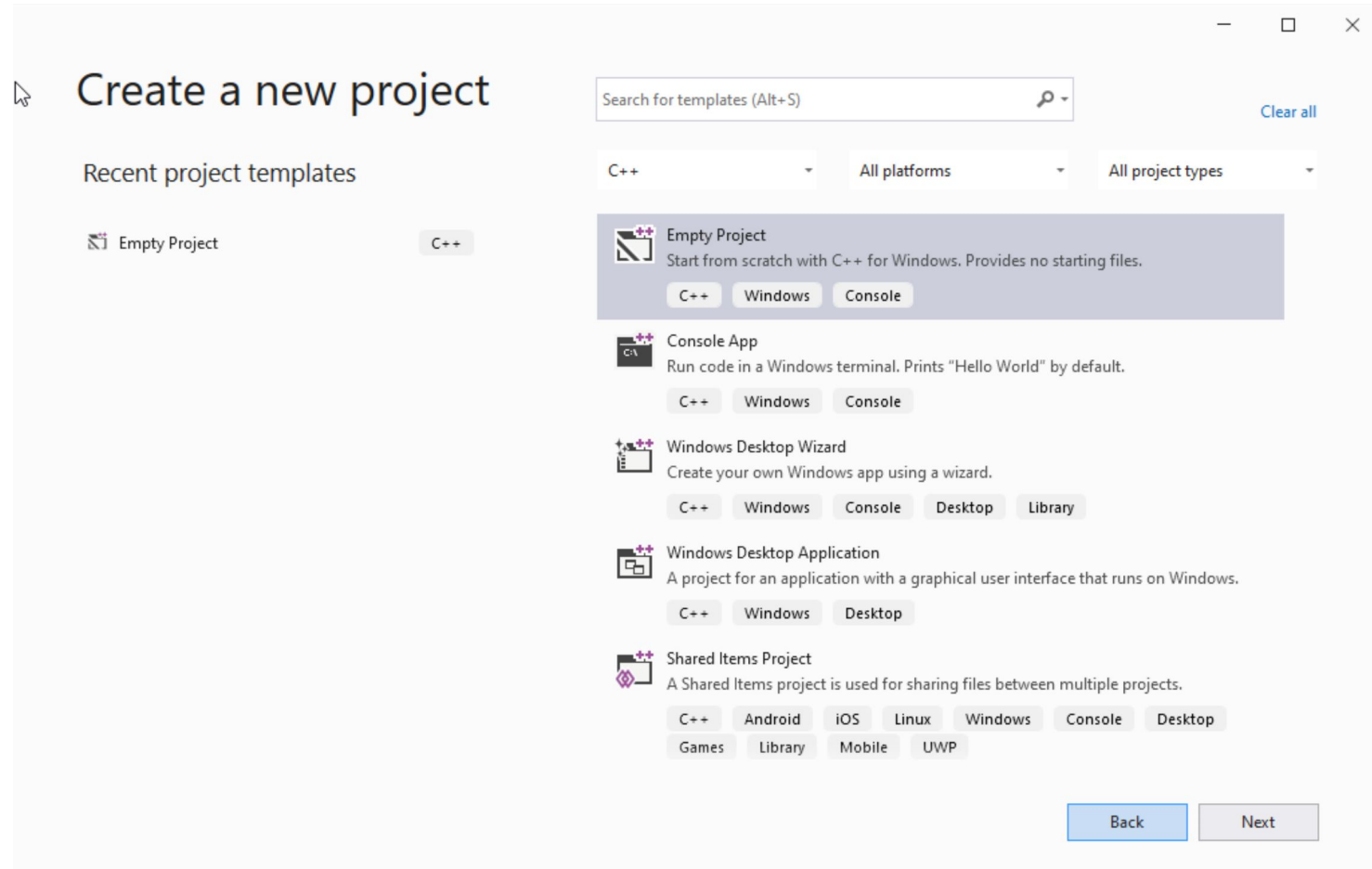
# Step 1: Create a project (1)

- (1) Start Visual Studio
- (2) Click Create a new Project



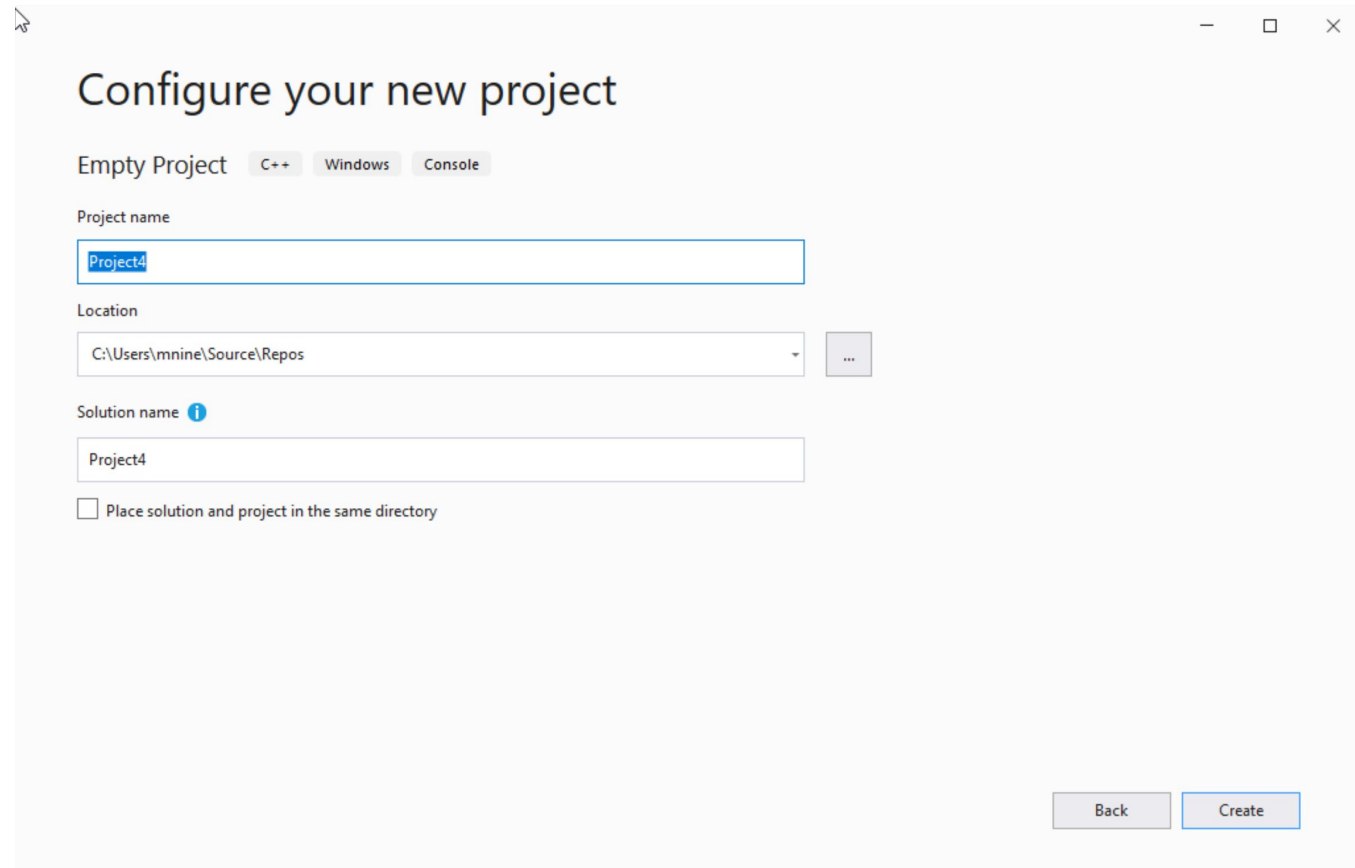
# Step 1: Create a project (2)

- (1) Select C++ as language
- (2) Select Empty Project
- (3) Click Next



# Step 1: Create a project (3)

- (1) You can change the project name as you like
- (1) Also, you can change the project location
- (2) Click Next



The screenshot shows the 'Configure your new project' dialog box in Visual Studio. The title bar includes standard window controls. The main heading is 'Configure your new project'. Below it, there are tabs for 'Empty Project', 'C++', 'Windows', and 'Console'. The 'Empty Project' tab is selected. The 'Project name' field contains 'Project4'. The 'Location' field shows the path 'C:\Users\mnine\Source\Repos' with a browse button ('...') to its right. The 'Solution name' field, which has an information icon, also contains 'Project4'. At the bottom, there is a checkbox labeled 'Place solution and project in the same directory' which is currently unchecked. In the bottom right corner, there are 'Back' and 'Create' buttons.

# Step 1: Create a project (4)

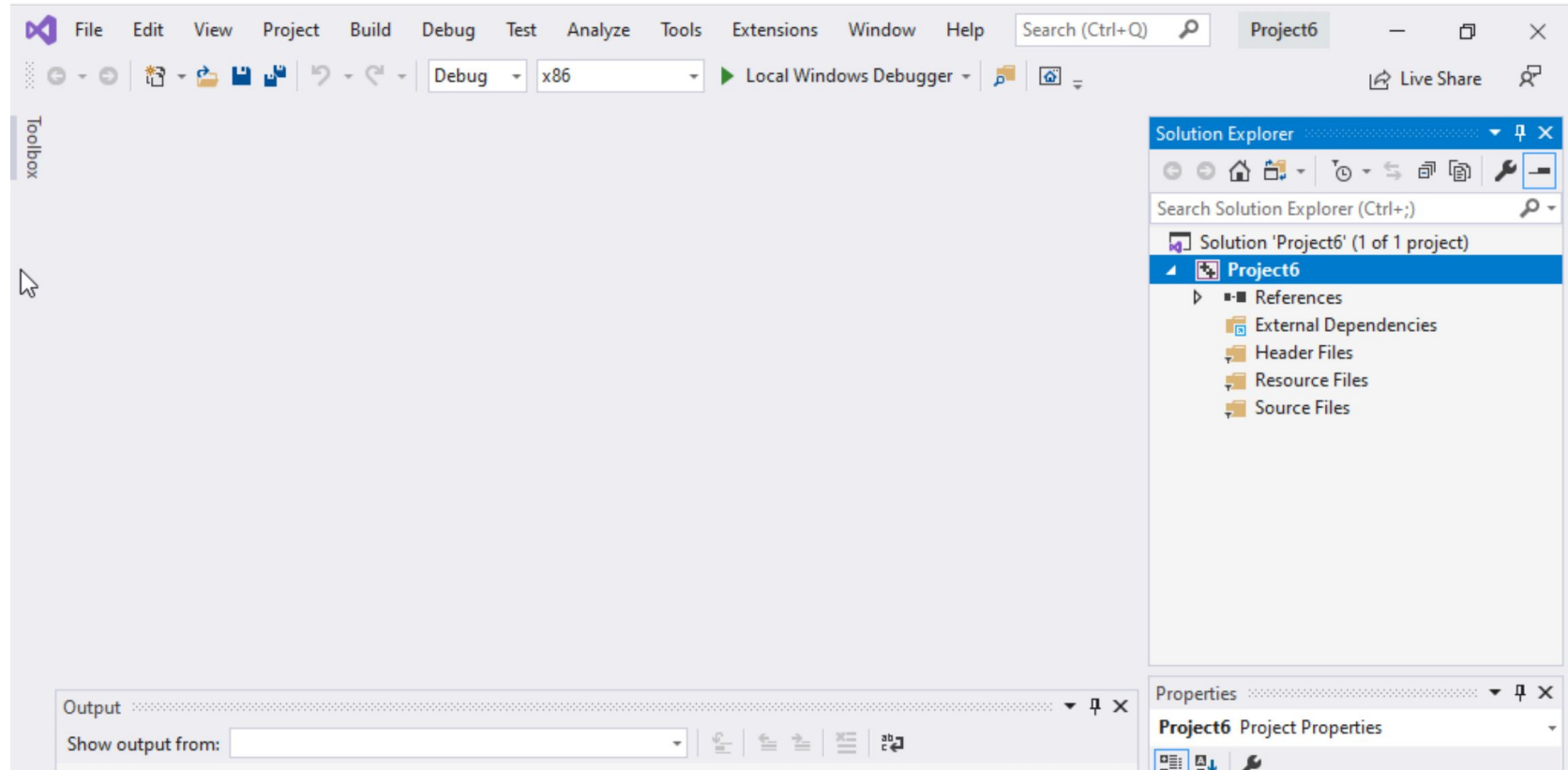
Delete the

Following folders:

- Header files

- Resources Files, and

- Source Files



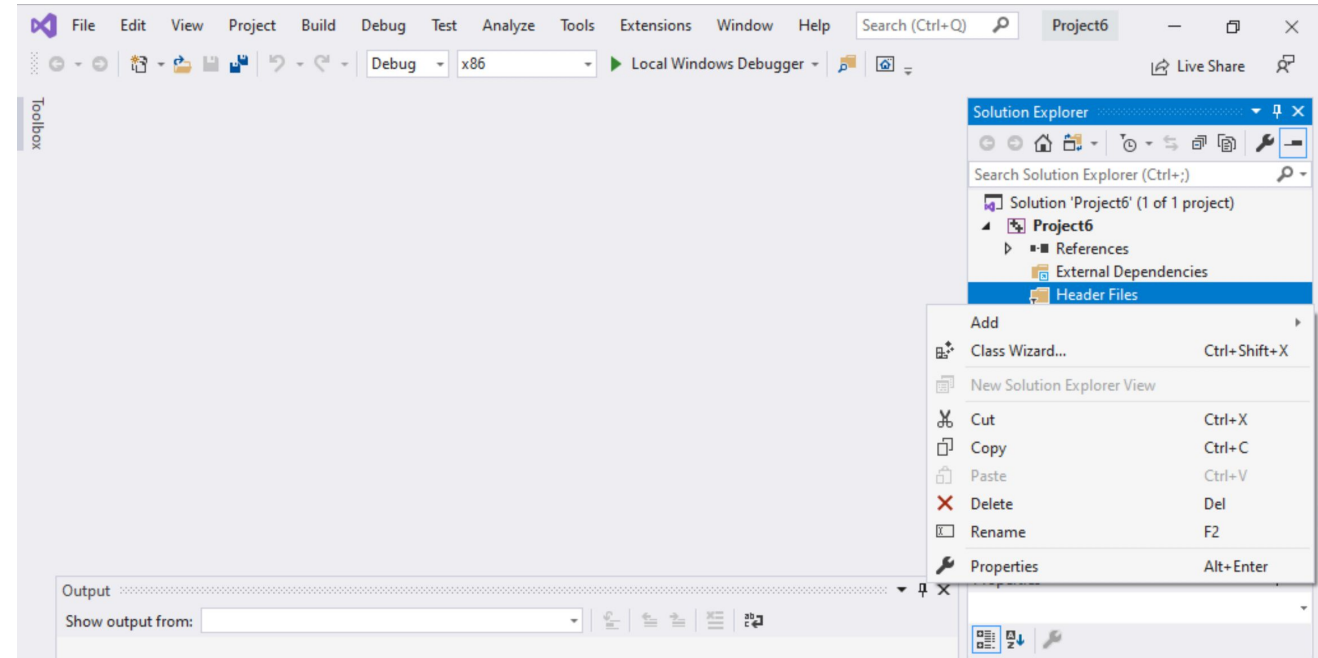
# Step 1: Create a project (5)

To delete :

Select the folders

Right click on it

Select delete



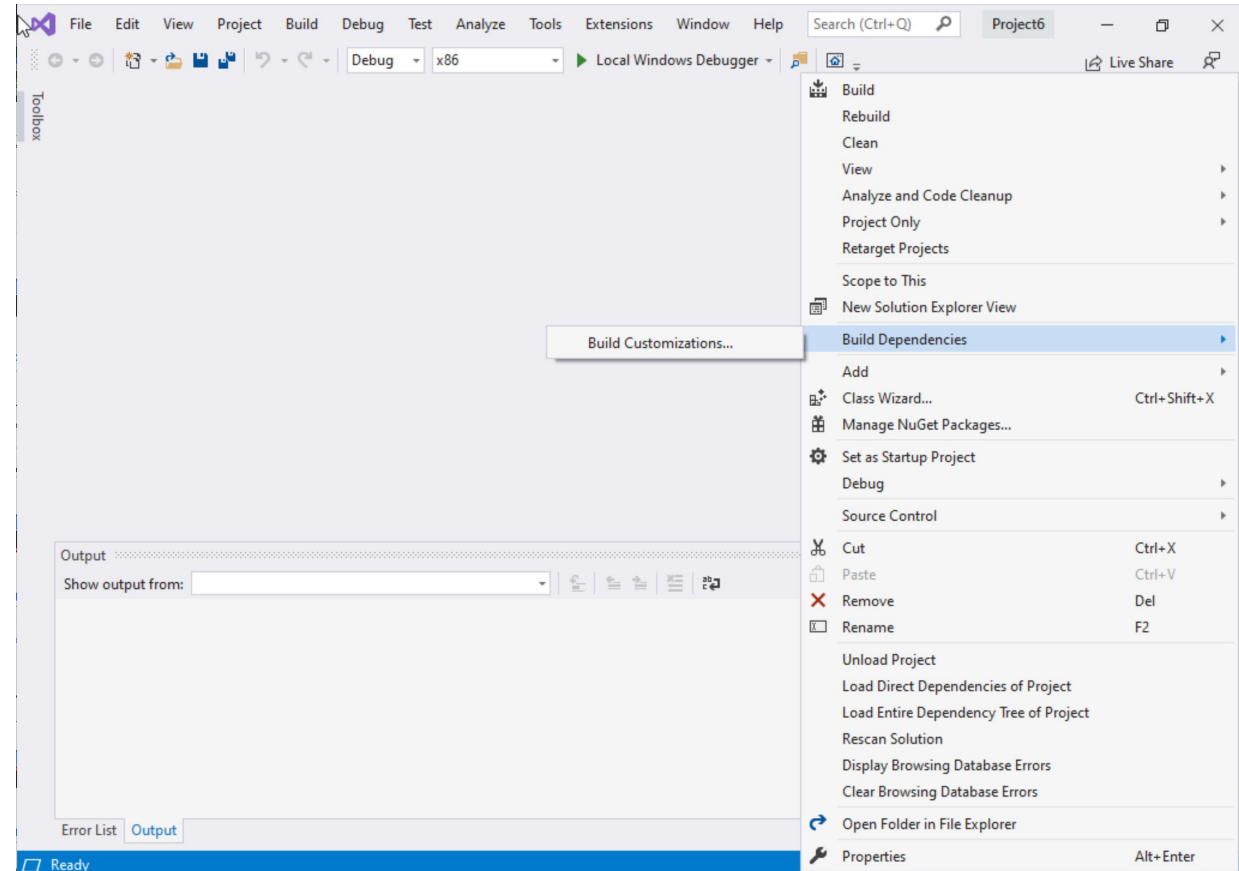
# Step 1: Create a project (6)

Select Project Name on solution explorer

Right click on it

Go to Build Dependencies

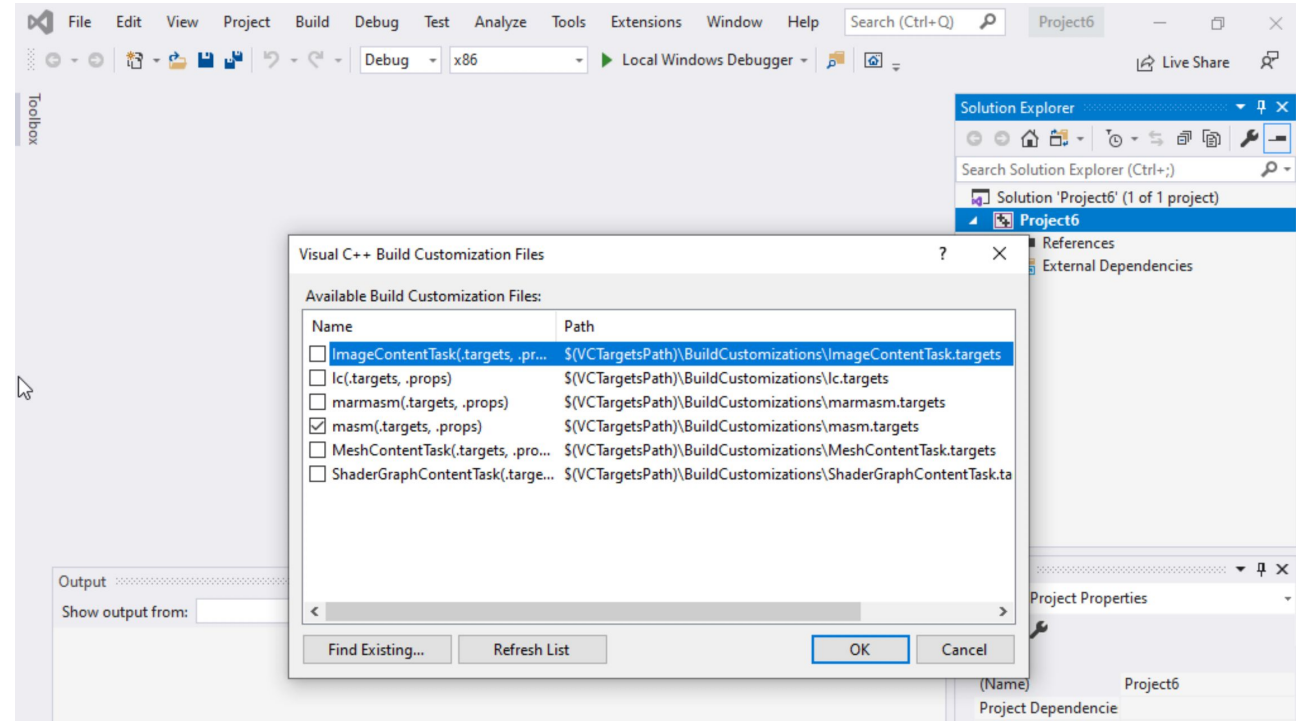
Click on Build Customizations



# Step 1: Create a project (7)

Select masm(.target, .props)

Click ok

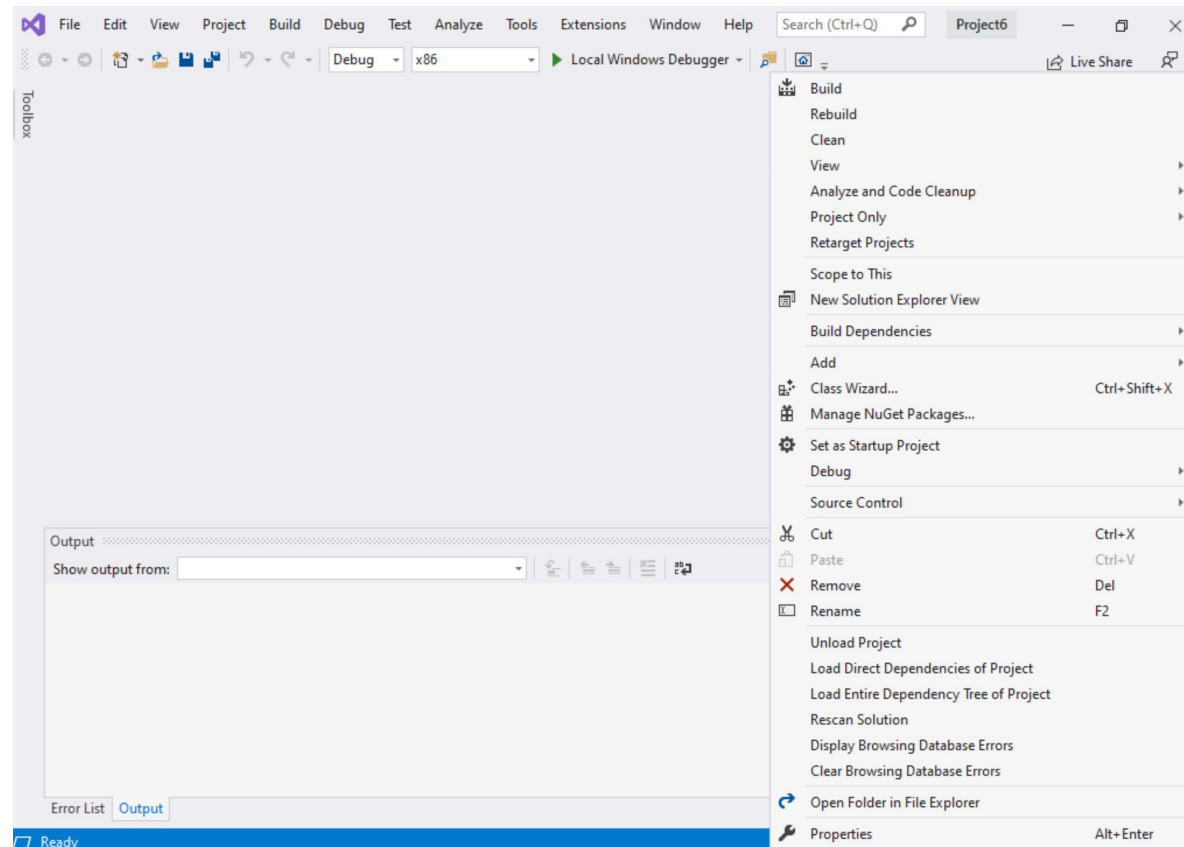




# Step 1: Create a project (8)

Right click on the Project name in the solution explorer

Click properties



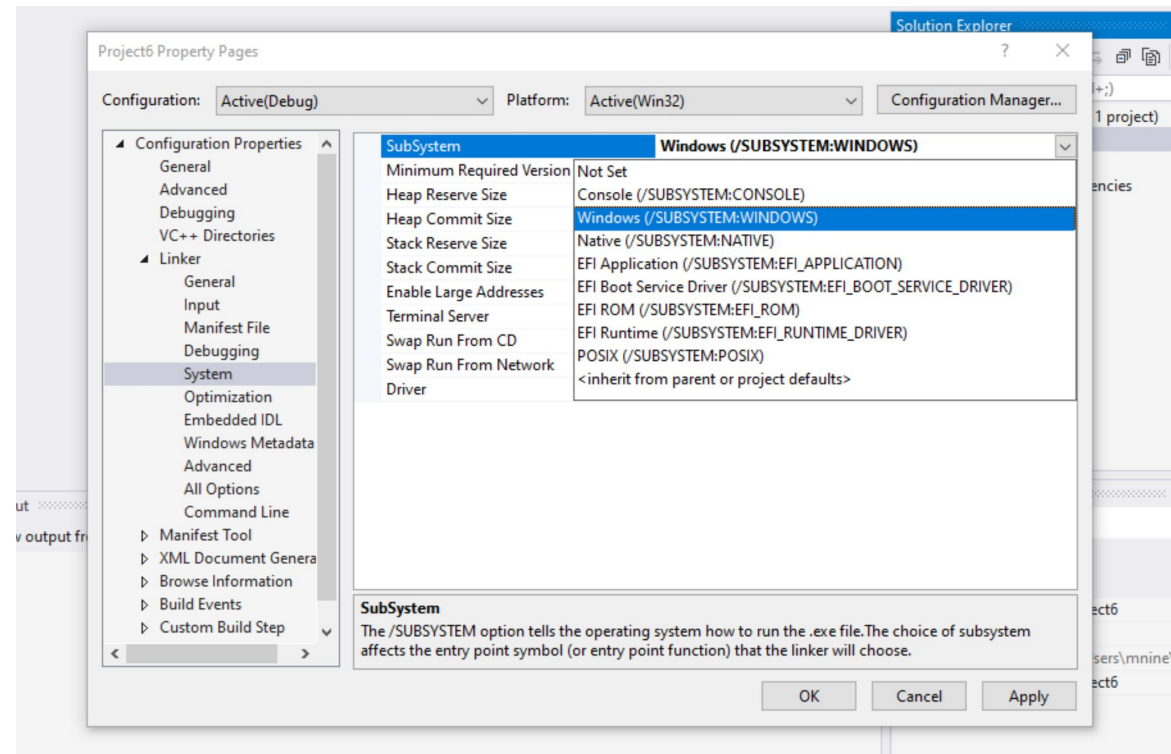
# Step 1: Create a project (9)

Expand the 'Linker'

Select 'System'

Select Windows(/SUBSYSTEM:WINDOWS)

Click OK



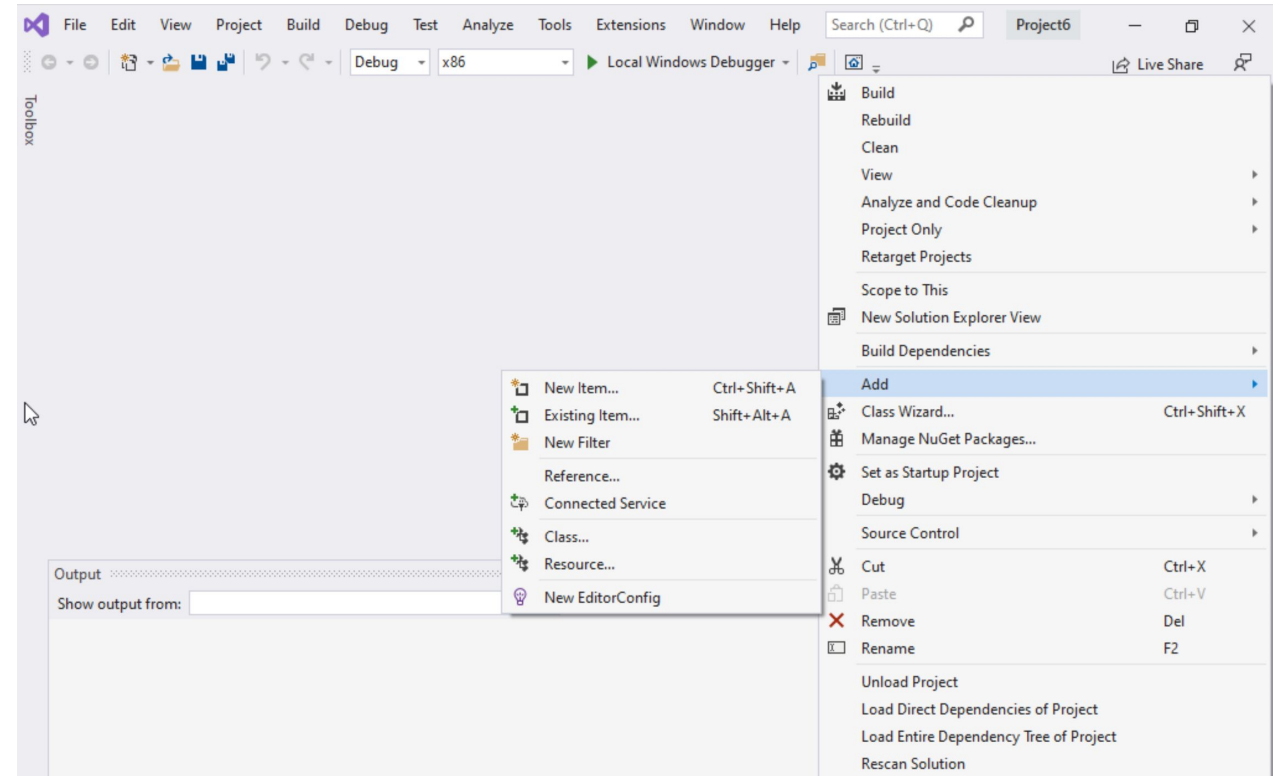
# Step 1: Create a project (10)

Select Project name on solution explorer

Right click on it

Expand Add

Choose New Item

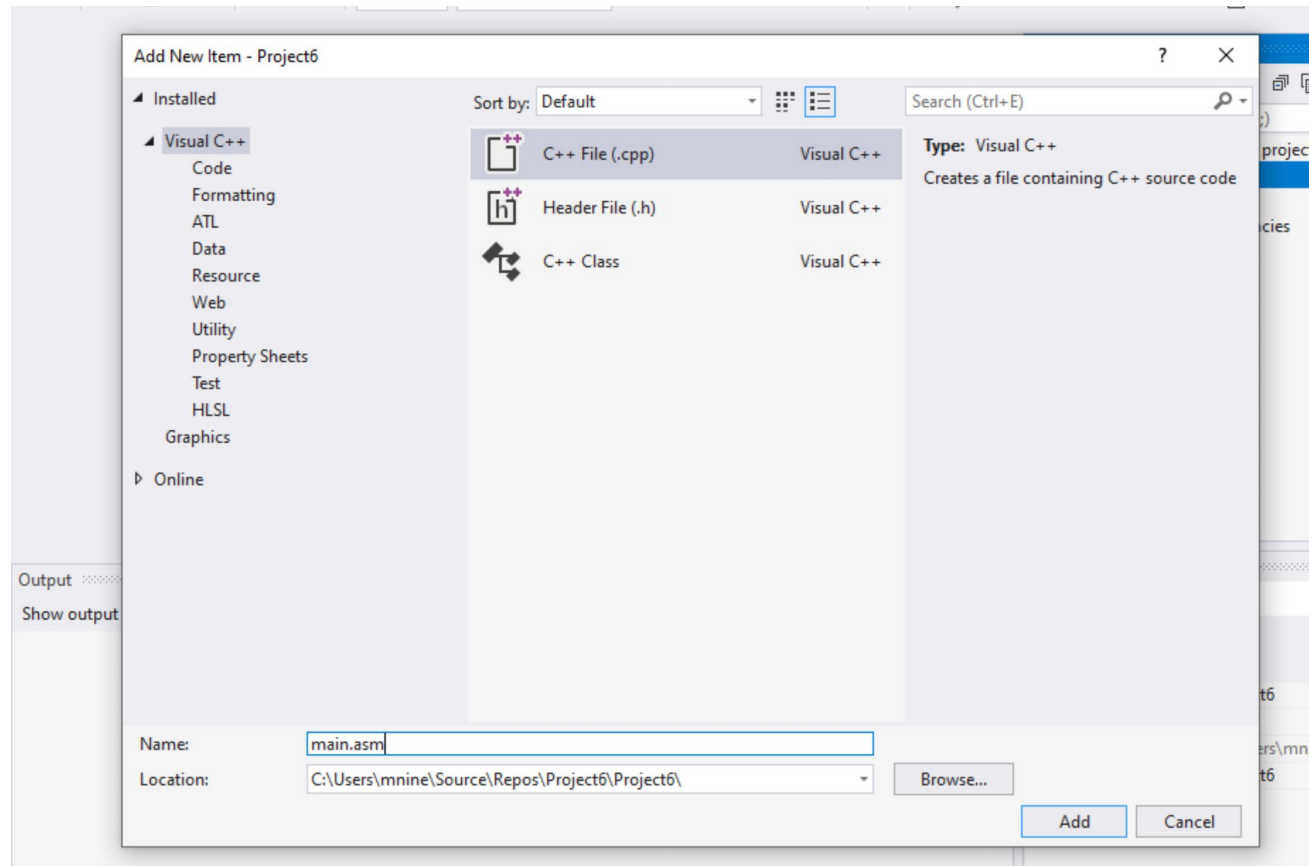


# Step 1: Create a project (11)

Select C++ File(.cpp)

Name: main.asm

Click Add



# Step 1: Create a project (12)

Select main.asm

Add your code

In the main.asm File.

