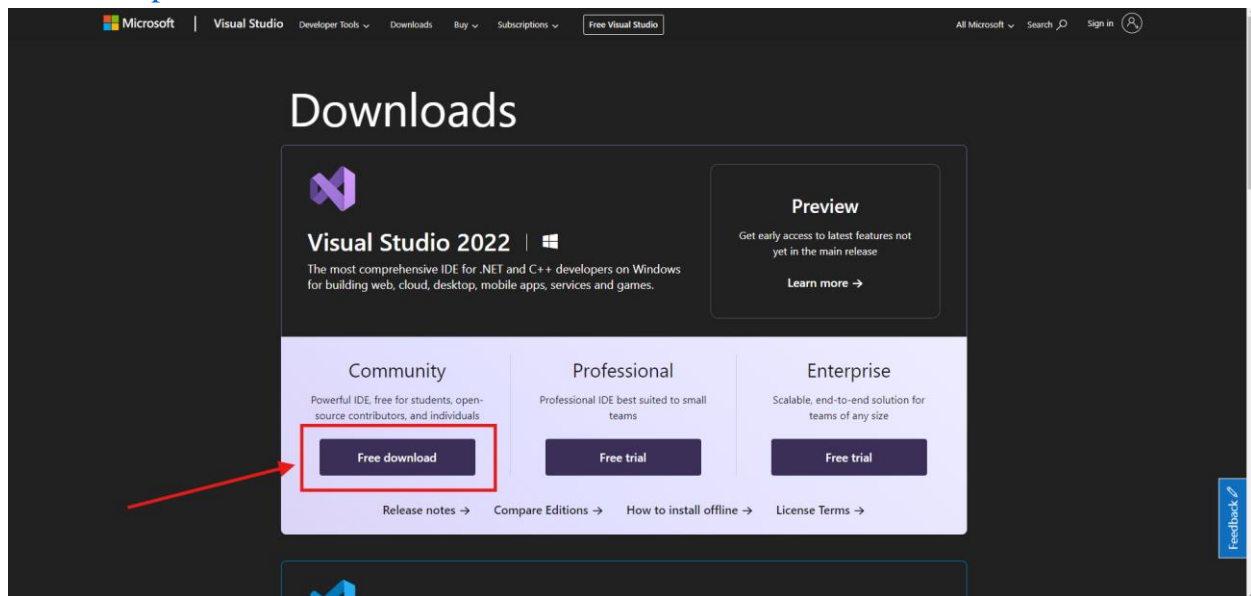


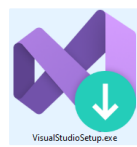
Practice Assignment 1

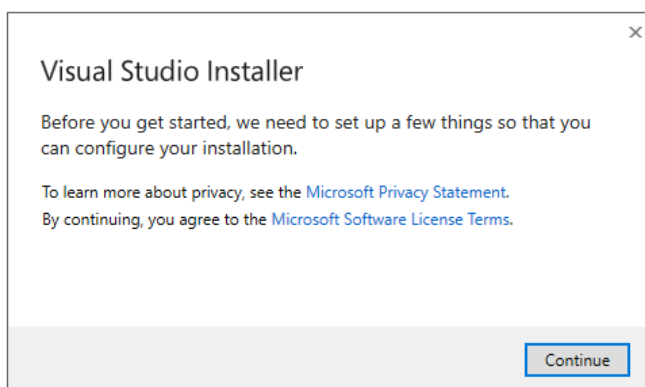
I. Prepare

1.1 Install Visual Studio Community 2022 (Skip if you have already installed it)

- Go to <https://visualstudio.microsoft.com/downloads/>



- After downloaded, open the file “VisualStudioSetup.exe”  for install.



- Click continue and wait for download the package for install.

After that click install button and wait until that finish.

Okay! After you finish the preparation, let's do the exercise below.

II. Exercise

Read

<https://csharpcodingguidelines.com>

<https://www.dofactory.com/csharp-coding-standards>

<http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>.

https://geosoft.no/javastyle.html#Naming_Conventions

Exercise 1: Based on the sources referenced above.

- Compare conventions of naming and formatting between common languages C# and Java.
- Create a detailed comparison with descriptions in words and save it as a Word or PDF file.

Exercise 2: Use the Visual Studio IDE to create a C# console project, paying attention to naming conventions and folder structure.

Exercise 3: Use the project created in Exercise 2, then add new source files to solve the following problem. Solve the problems effectively and pay attention to naming conventions.

Problem 1: *Given two sorted arrays `nums1` and `nums2` of size m and n respectively, return the median of the two sorted arrays.*

The overall run time complexity should be $O(\log(m+n))$.

Example 1:

Input: `nums1 = [1,3]`, `nums2 = [2]`

Output: 2.00000

Explanation: merged array = `[1,2,3]` and median is 2.

Example 2:

Input: `nums1 = [1,2]`, `nums2 = [3,4]`

Output: 2.50000

Explanation: merged array = `[1,2,3,4]` and median is $(2 + 3) / 2 = 2.5$.

Constraints:

`nums1.length == m`

`nums2.length == n`

$0 \leq m \leq 1000$

$0 \leq n \leq 1000$

$1 \leq m + n \leq 2000$

$-106 \leq \text{nums1}[i], \text{nums2}[i] \leq 106$

Problem 2: Given two integers **dividend** and **divisor**, divide two integers **without** using multiplication, division, and mod operator.

The integer division should truncate toward zero, which means losing its fractional part. For example, **8.345** would be truncated to **8**, and **-2.7335** would be truncated to **-2**.

Return the quotient after dividing dividend by divisor.

Note: Assume we are dealing with an environment that could only store integers within the **32-bit** signed integer range: $[-2^{31}, 2^{31} - 1]$. For this problem, if the quotient is **strictly greater than** $2^{31} - 1$, then return $2^{31} - 1$, and if the quotient is **strictly less than** -2^{31} , then return -2^{31} .

Example 1:

Input: dividend = 10, divisor = 3

Output: 3

Explanation: $10/3 = 3.33333..$ which is truncated to 3.

Example 2:

Input: dividend = 7, divisor = -3

Output: -2

Explanation: $7/-3 = -2.33333..$ which is truncated to -2.

Constraints:

- $-2^{31} \leq \text{dividend}$, $\text{divisor} \leq 2^{31} - 1$
- $\text{divisor} \neq 0$

Problem 3: Given an **m x n** grid of characters **board** and a string **word**, return **true** if **word** exists in the grid.

The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are **horizontally** or **vertically** neighboring. The same letter cell may not be used more than once.

Example 1:

A	B	C	E
S	F	C	S
A	D	E	E

Input: board = [["A","B","C","E"], ["S","F","C","S"], ["A","D","E","E"]],
word = "ABCCED"

Output: true

Example 2:

A	B	C	E
S	F	C	S
A	D	E	E

Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"

Output: true

Example 3:

A	B	C	E
S	F	C	S
A	D	E	E

Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"

Output: false

Constraints:

- $m == \text{board.length}$
- $n = \text{board}[i].\text{length}$
- $1 \leq m, n \leq 6$
- $1 \leq \text{word.length} \leq 15$
- board and word consists of only lowercase and uppercase English letters.