



# GitHub Classroom Auto-grading

In C programming language

09-2025

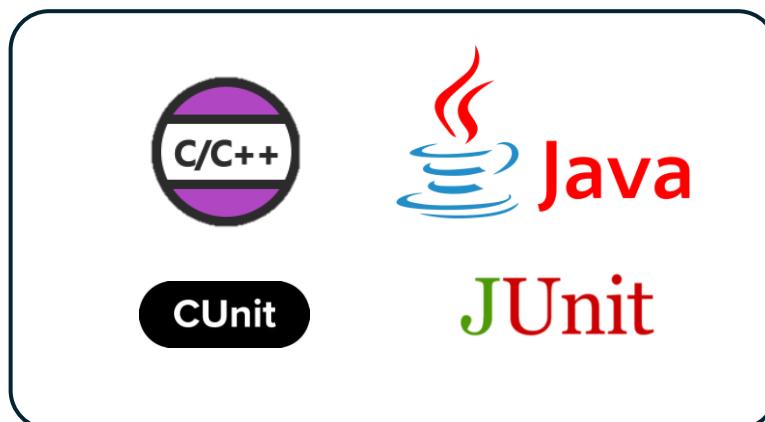
Nhóm thực hiện

- Thân Hoàng Lộc
- Trần Quang Bình
- Trần Văn Huy
- Trần Anh Tuấn

# Mục tiêu

Kiểm tra nội dung và Chấm điểm tự động  
bài làm của sinh viên  
đã nộp trên phần mềm GitHub Classroom

Nhóm 1



Nhóm 2



Nhóm 3



# Nội dung Nhóm 1 (C/C++)

1. Giới thiệu
2. MinGW, CUnit
3. VS Code Configure C/C++ Compiler – tasks.json
4. VS Code Configure C/C++ Debugger – launch.json
5. Application & Unit testing
6. Makefile & Build command & Test runner
7. Auto-grading run command
8. Auto-grading Reporter



# Giới thiệu GitHub Classroom Auto-grading

**Tính năng Auto-grading, để kiểm tra và chấm điểm tự động  
nội dung bài làm của sinh viên đã nộp trên GitHub Classroom**

## Giảng viên

- Cấu hình chế độ kiểm tra:  
ngay lập tức hay định kỳ
- Xem báo cáo tổng quát:  
tình hình nộp bài và  
tình trạng bài nộp ✓ Passed hay ✗ Failed

## Sinh viên

- Xem và biết kết quả:  
bài đã nộp ✓ Passed hay ✗ Failed



# Giới thiệu GitHub Classroom Auto-grading

Có 3 nhóm phương pháp Auto-grading

**run command tests**



**JUnit**

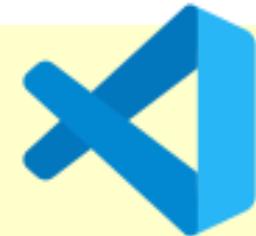
**python tests**

**py**t**est**

**Input/Output tests**



**BASH**  
&  
Shell Scripts



# Giới thiệu GitHub Classroom Auto-grading

Nội dung bài làm của sinh viên, và **ĐÃ NỘP** (commit + push) thành công

Lỗi biên dịch (build error)  
Không có hàm "max" trong thư viện "find\_max.h"

Lỗi logic (run-time error)  
Thuật toán "c = max" chưa chính xác

Nội dung bài làm của sinh viên, và **ĐÃ NỘP** (commit + push) thành công

# Giới thiệu GitHub Classroom Auto-grading

Classrooms / COM108-UDPM-classroom-class-SU25-01 / LAB5-Bai1-autograding

## LAB5-Bai1-autograding

Starter code from [COM108-UDPM/com108-udpm-classroom-class-su25-01-lab5-bai1-autograding-asm-autograding-com108-lab5b1](https://classroom.github.com/a/ML7IdGSn)

Individual assignment Due Sep 30, 2025, 16:00 UTC Active Sync assignments https://classroom.github.com/a/ML7IdGSn Run tests Edit Download

**Assignment Details**

Students total 2	Accepted assignments 2	Assignment submissions 2	Passed students 1
2 Rostered 0 Added students	2 Students	2 Submitted 0 Not submitted	1/2 Passed <div style="width: 50%; height: 10px; background-color: green; margin-top: 5px;"></div>

Filters Search for an assianment Filter by unlinked accounts Filter by accepted Filter by passing Sort

### Classroom roster

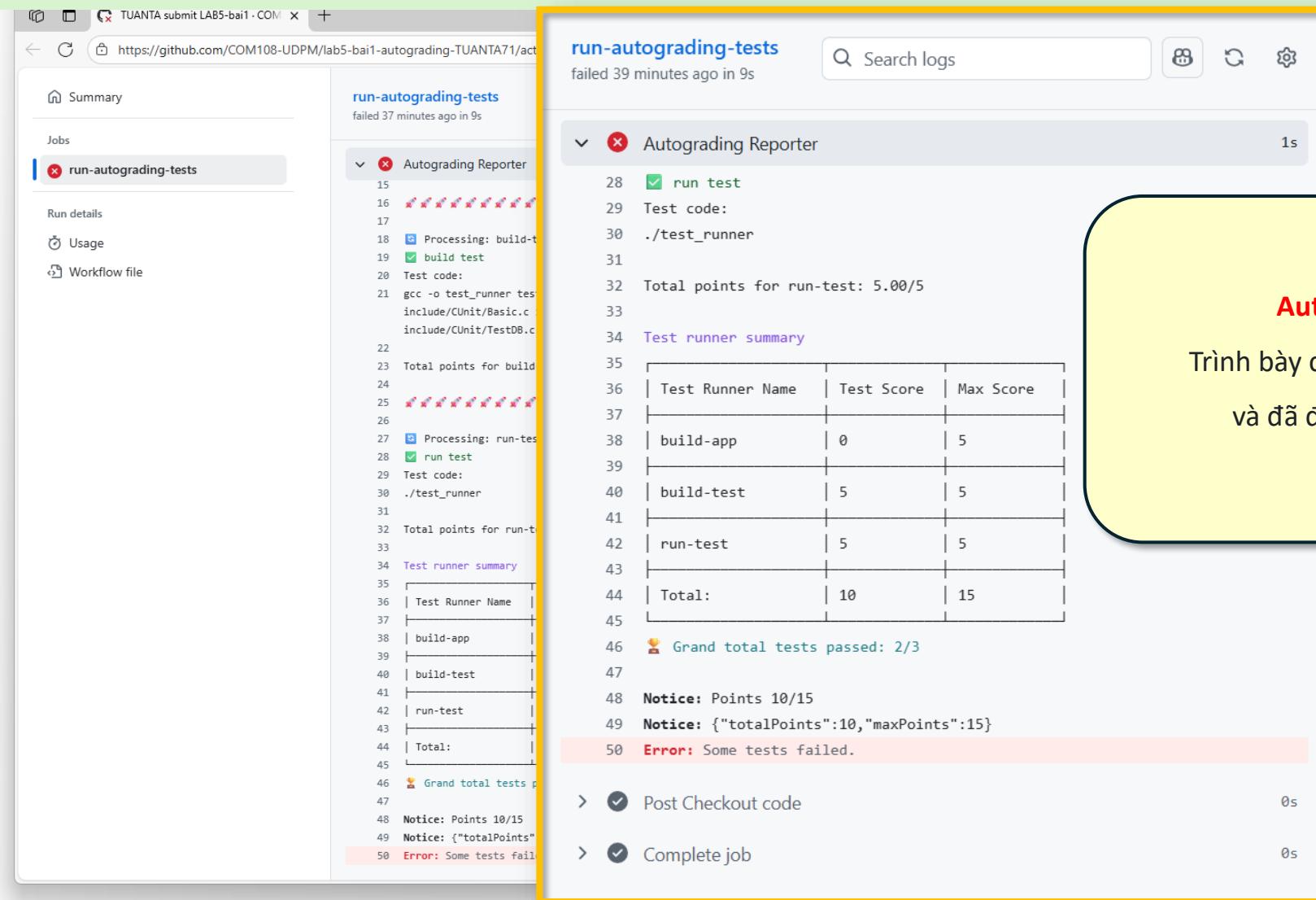
 TuanAPC	Submitted	<a href="#">Latest commit yesterday</a> ✓	3 commits	<div style="width: 150px; background-color: green; height: 10px; display: inline-block;"></div> 15/15
 TuanTA71	Submitted	<a href="#">Latest commit 22 minutes ago</a> ✗	5 commits	<div style="width: 150px; background-color: red; height: 10px; display: inline-block;"></div> 10/15

Kết quả được kiểm tra tự động

Thống kê tổng quát và chi tiết từng bài  
đã nộp của từng sinh viên

✓ Passed      ✗ Failed

# Giới thiệu GitHub Classroom Auto-grading



The screenshot shows a GitHub Classroom interface for a repository named 'TUANTA submit LAB5-bai1'. The main window displays a 'run-autograding-tests' job that failed 37 minutes ago. The 'Autograde Reporter' section is highlighted with a yellow box and expanded in a callout bubble.

**Autograde Reporter**

```

28  ✓ run test
29 Test code:
30 ./test_runner
31
32 Total points for run-test: 5.00/5
33
34 Test runner summary
35
36 | Test Runner Name | Test Score | Max Score |
37 |                 |           |           |
38 | build-app       |     0     |      5      |
39 |                 |           |           |
40 | build-test      |     5     |      5      |
41 |                 |           |           |
42 | run-test        |     5     |      5      |
43 |                 |           |           |
44 | Total:          |    10    |     15     |
45
46 🎉 Grand total tests passed: 2/3
47
48 Notice: Points 10/15
49 Notice: {"totalPoints":10,"maxPoints":15}
50 Error: Some tests failed.

```

The callout bubble contains the following text:

**Auto-grading Reporter**

Trình bày chi tiết các nội dung đã test  
và đã được thực hiện **tự động**

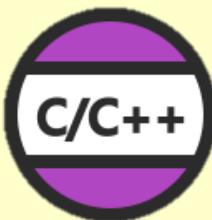
# Giới thiệu GitHub Classroom Auto-grading

The screenshot shows a GitHub Classroom repository for a lab assignment. The main commit message is 'TUANTA submit LAB5-bai1'. A yellow hand icon points to this message, with a black oval containing the word 'failure' overlaid. Below the commit message, there is a yellow box highlighting the status bar which says 'All checks have failed' and '1 failing check'. A red box highlights the specific failing check: 'Autograding Tests / run-autograding-tests (push) Failing after 9s'. To the right, a yellow callout bubble contains the text 'Auto-grading Reporter' and 'Sinh viên biết được ngay kết quả và xem chi tiết lỗi' (Student knows the result immediately and can view detailed errors). It also includes icons for 'Passed' (green checkmark) and 'Failed' (red X).

**Auto-grading Reporter**

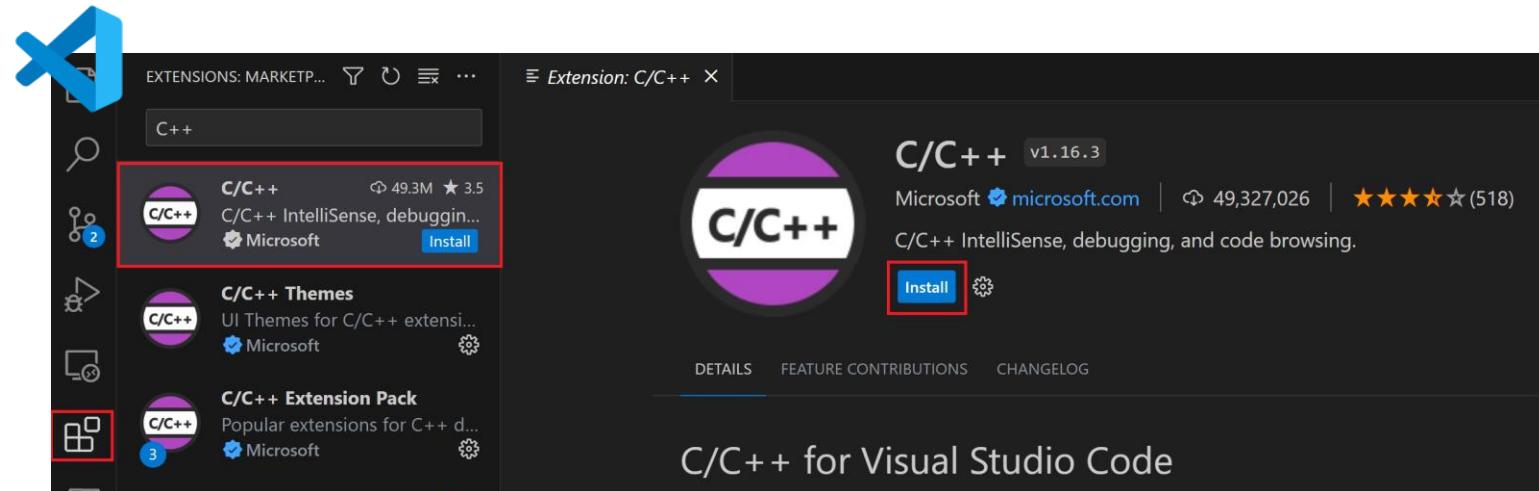
Sinh viên biết được ngay kết quả  
và xem chi tiết lỗi

Passed Failed



# MinGW: *MinGW-w64 (64bit, 32bit), MinGW (32bit)*

*Tham khảo cài đặt MinGW-w64, tại [https://code.visualstudio.com/docs/cpp/config-mingw#\\_prerequisites](https://code.visualstudio.com/docs/cpp/config-mingw#_prerequisites)*



**CUnit**

# A unit testing framework for C

Tham khảo tại <https://cunit.sourceforge.net>

Dùng thư viện CUnit để thực hiện 2 việc sau

**1**

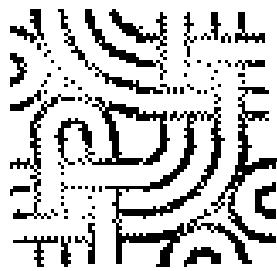
Tạo các test case  
cho chương trình  
(lab / assignment)

**2**

Tạo các auto-grading  
run command  
**mỗi khi** repository commit



# Các thư viện khác



CppUnit Testing Library

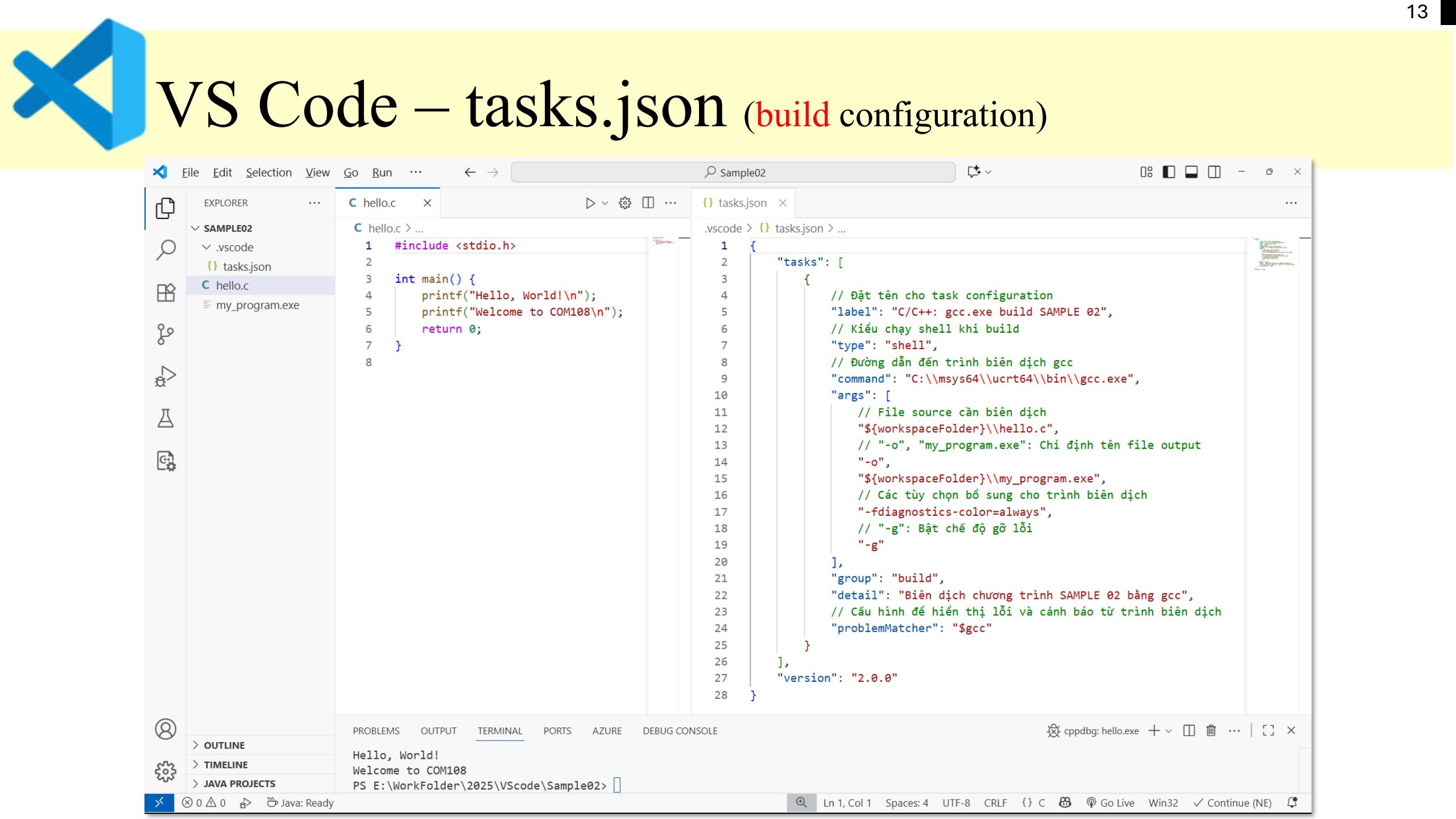
<https://cppunit.sourceforge.net/doc/1.8.0/index.html>



C# Dev Kit

[https://marketplace.visualstudio.com/items?  
itemName=ms-dotnettools.csdevkit](https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csdevkit)

# VS Code – tasks.json (build configuration)



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project named "SAMPLE02" containing ".vscode", "tasks.json", "hello.c", and "my\_program.exe".
- Code Editor:** Displays the content of "hello.c" which prints "Hello, World!" and "Welcome to COM108".
- Code Editor (Right):** Displays the "tasks.json" file, which defines a build task for "hello.c" using the "shell" type and "gcc" command.
- Terminal:** Shows the output of the build process: "Hello, World!", "Welcome to COM108", and "PS E:\WorkFolder\2025\VScode\Sample02>".
- Status Bar:** Shows "Java: Ready" and other standard status bar information.

```

{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "C/C++: gcc.exe build SAMPLE 02",
      "type": "shell",
      "command": "C:\\msys64\\ucrt64\\bin\\gcc.exe",
      "args": [
        "${workspaceFolder}\\\\hello.c",
        "-o",
        "${workspaceFolder}\\\\my_program.exe",
        "-fdiagnostics-color=always",
        "-g"
      ],
      "group": "build",
      "detail": "Biên dịch chương trình SAMPLE 02 bằng gcc",
      "problemMatcher": "$gcc"
    }
  ]
}
  
```

# VS Code – tasks.json (build configuration)

The screenshot shows the VS Code interface with the title "VS Code – tasks.json (build configuration)". The left sidebar displays a file tree with a project named "SAMPLE02" containing ".vscode", "tasks.json", "hello.c", and "my\_program.exe". The "EXPLORER" icon is highlighted. A yellow arrow points to the first item in the "Select the build task to run" dropdown, which is "C/C++: gcc.exe build SAMPLE 02". Below it, the text "Biên dịch chương trình SAMPLE 02 bằng gcc" is displayed in Vietnamese. The "recently used tasks" section shows "C/C++: cl.exe build active file" and "compiler: cl.exe". The "detected tasks" section shows "C/C++: gcc.exe build active file" and "compiler: C:\msys64\ucrt64\bin\gcc.exe".

**Build**    **ctrl**   **shift**   **B**

**Run**    **ctrl**   **F5**

```
13 // "-o", "my_program.exe": Chỉ định tên file output
14 "-o",
15 "${workspaceFolder}\my_program.exe",
16 // Các tùy chọn bổ sung cho trình biên dịch
17 "-fdiagnostics-color=always",
18 // "-g": Bật chế độ gõ lỗi
19 "-g"
20 ],
21 "group": "build",
22 "detail": "Biên dịch chương trình SAMPLE 02 bằng gcc",
23 // Cấu hình để hiển thị lỗi và cảnh báo từ trình biên dịch
24 "problemMatcher": "$gcc"
25 }
26 ],
27 "version": "2.0.0"
```

PROBLEMS   OUTPUT   TERMINAL   PORTS   AZURE   DEBUG CONSOLE

Hello, World!  
Welcome to COM108  
PS E:\WorkFolder\2025\VScode\Sample02>

cppdbg: hello.exe + ... | [] X

Ln 1, Col 1   Spaces: 4   UTF-8   CRLF   {} C   Go Live   Win32   Continue (NE)   🔍



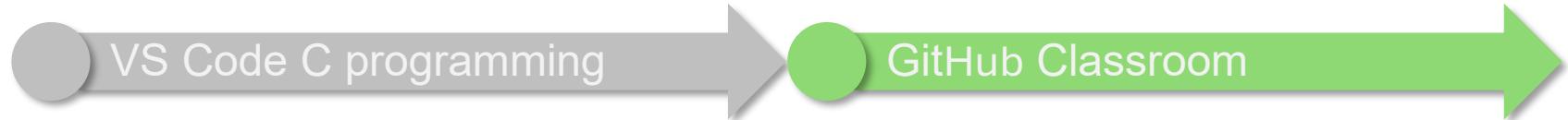
# VS Code – tasks.json (build configuration)

```
.vscode > {} tasks.json > ...
1  {
2    "tasks": [
3      {
4        // Đặt tên cho task configuration
5        "label": "C/C++: gcc.exe build SAMPLE 02",
6        // Kiểu chạy shell khi build
7        "type": "shell",
8        // Đường dẫn đến trình biên dịch gcc
9        "command": "C:\\msys64\\ucrt64\\bin\\gcc.exe",
10       "args": [
11         // File source cần biên dịch
12         "${workspaceFolder}\\\\hello.c",
13         // "-o", "my_program.exe": Chỉ định tên file c
14         "-o",
15         "${workspaceFolder}\\\\my_program.exe",
16         // Các tùy chọn bổ sung cho trình biên dịch
17         "-fdiagnostics-color=always",
18         // "-g": Bật chế độ gỡ lỗi
19         "-g"
20       ],
21       "group": "build",
22       "detail": "Biên dịch chương trình SAMPLE 02 bằng gcc",
23       // Cấu hình để hiển thị lỗi và cảnh báo từ trình biên dịch
24       "problemMatcher": "$gcc"
25     ],
26   },
27   "version": "2.0.0"
28 }
```

The screenshot shows the VS Code interface with the terminal tab active. The terminal window title is "cmd". The command entered is "\${workspaceFolder}> gcc -o hello.c my\_program.exe". The entire command line is highlighted with a red rectangle. A yellow arrow points from the "command" field in the tasks.json configuration to the "gcc" part of the terminal command. A red arrow points from the "my\_program.exe" part of the terminal command to the "my\_program.exe" part in the tasks.json configuration.

Đây là “run command”, nhóm build

# Tóm tắt



- ✓ 1. Cài đặt VS Code
  - ✓ 2. Cài đặt MinGW
  - ✓ 3. Build configuration – tasks.json
  - ✓ 4. Simple build run command
- Create classroom (Tạo lớp)
  - Create assignment, starter code
  - Add auto-grading



# VS Code – Hello-auto-grading

**Bước 1 : Tạo Project (starter code, build configuration)**

The screenshot shows the VS Code interface with the following components:

- EXPLORER**: Shows a project structure with a folder named "FPOLY..." containing ".vscode" and "tasks.json". A file named "hello.c" is selected.
- Editor**: Displays the content of "hello.c" which contains a simple C program to print "Hello, World!".
- tasks.json**: Shows the build configuration for the project. The "tasks" array contains one task for building with gcc. The task is labeled "C/C++: gcc.exe build HELLO auto-grading" and uses a shell type. It specifies the command as "C:\msys64\ucrt64\bin\gcc.exe" and provides arguments for the source file ("\${workspaceFolder}\hello.c") and the output file ("\${workspaceFolder}\my\_program.exe"). It also includes diagnostic options like "-fdiagnostics-color=always" and "-g" for debugging.

**Starter Code**

```

1 #include <stdio.h>
2
3 // Viết một chương trình C
4 // để in ra các từ "Hello, World!"
5
6 // Ví dụ như sau
7 // main() {
8
9 //     // In ra "Hello, World!"
10
11 // } // Kết thúc chương trình
12
13 // }
14
15
16

```

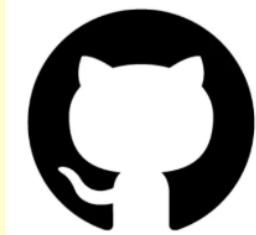
**Build Configuration**

```

1 {
2     "tasks": [
3         {
4             // Đặt tên cho task configuration
5             "label": "C/C++: gcc.exe build HELLO auto-grading",
6             // Kiểu chạy shell khi build
7             "type": "shell",
8             // Đường dẫn đến trình biên dịch gcc
9             "command": "C:\\msys64\\ucrt64\\bin\\gcc.exe",
10            "args": [
11                // File source cần biên dịch
12                "${workspaceFolder}\\hello.c",
13                // "-o", "my_program.exe": Chỉ định tên file output
14                "-o",
15                "${workspaceFolder}\\my_program.exe",
16                // Các tùy chọn bổ sung cho trình biên dịch
17                "-fdiagnostics-color=always",
18                // "-g": Bật chế độ gỡ lỗi
19                "-g"
20            ],
21            "group": "build",
22            "detail": "Biên dịch chương trình HELLO auto-grading bằng gcc",
23            // Cấu hình để hiển thị lỗi và cảnh báo từ trình biên dịch
24            "problemMatcher": "$gcc"
25        }
26    ],
27    "version": "2.0.0"
28 }

```

Java: Ready



# GitHub – Hello-auto-grading repository

## Bước 2 – Template repository

- Tạo repository
- Commit & Push Project
- Bật Template Repository

The screenshot shows a GitHub repository page for 'vnTranTuan/FPOLY-Hello-auto-grading'. A red box highlights the title 'Template Repository' above the repository details. The repository has 1 branch and 0 tags. It contains a '.vscode' folder and a 'hello.c' file, both committed 7 minutes ago by 'vnTranTuan' with the message 'Hello auto-grading'. Below the files is a 'README' section with a 'Add a README' button. The repository has 0 stars, 0 forks, and 0 watching. It also has 0 releases published and 0 packages published. The 'Languages' section shows 100% C usage.

# Tạo Assignment Auto-grading

The screenshot shows the 'New assignment' creation page in GitHub Classroom. The URL in the browser is [https://classroom.github.com/classrooms/227212486-com108-udpm-classroom-class-su25-01/new\\_assignments/new](https://classroom.github.com/classrooms/227212486-com108-udpm-classroom-class-su25-01/new_assignments/new). The page title is 'Let's set up the basics for your assignment.' A sidebar on the left lists 'Assignment creation steps': 'Assignment basics' (selected), 'Starter code and environment', and 'Grading and feedback'. The main form on the right contains fields for 'Assignment title \*' (set to 'LAB-hello-auto-grading'), 'Deadline' (set to '10/31/2025 11:00 PM'), and 'Individual or group assignment' (set to 'Individual assignment'). There is also a checkbox for 'This is a cutoff date' with a note explaining it. At the bottom are 'Cancel' and 'Continue' buttons.

Bước 3 – New assignment

- a. New assignment
- b. Assignment basics

# Tạo Assignment Auto-grading

The screenshot shows the GitHub Classroom interface for creating a new assignment. The URL in the browser is [https://classroom.github.com/classrooms/227212486-com108-udpm-classroom-class-su25-01/new\\_assignments/continue?current\\_step=1](https://classroom.github.com/classrooms/227212486-com108-udpm-classroom-class-su25-01/new_assignments/continue?current_step=1). The page title is "Classroom". The main heading says "Add your starter code and choose an optional online IDE.". On the left, there's a sidebar titled "Assignment creation steps" with three options: "Assignment basics" (marked with a green checkmark), "Starter code and environment" (which is selected and highlighted in blue), and "Grading and feedback". The main content area starts with "Add a repository to give students starter code". It explains that the assignment will be created with empty student repositories if no starter code is added. It also notes that the starter code repository must be a template; a copy will be created in the organization. It mentions that accepted assignments will be forks of the copy created in the organization. Below this, there's a search bar labeled "Find a GitHub repository" containing "FPOLY-Hello-auto-", and a dropdown menu showing a result: "vnTranTuan/FPOLY-Hello-auto-grading". Further down, there's a section for "Repository visibility" with two radio button options: "Private" (selected) and "Public". A note under "Private" says "Private repositories will only be visible to the student and the classroom owners." A note under "Public" says "Public repositories will be visible to everyone, including other students." At the bottom, there's a checkbox for "Grant students admin access to their repository".

Bước 4 –  
Starter code and environment  
Select GitHub repository

Add your starter code and choose an optional online IDE.

Assignment creation steps

✓ Assignment basics

Starter code and environment

Grading and feedback

Add a repository to give students starter code

Your assignment will be created with empty student repositories if you don't add starter code.

The starter code repository must be a template; a copy will be created in your organization.

Accepted assignments will be forks of the copy created in your organization. If you make changes to the copy created in your organization, you can press the "Sync assignments" button on the assignment dashboard to open Pull Requests in all student assignment repositories with the changes.

Find a GitHub repository

FPOLY-Hello-auto-

vnTranTuan/FPOLY-Hello-auto-grading

Repository visibility

Private

Private repositories will only be visible to the student and the classroom owners.

Public

Public repositories will be visible to everyone, including other students.

Grant students admin access to their repository

# Tạo Assignment Auto-grading

The screenshot shows the GitHub Classroom interface for setting up auto-grading. The main window displays the 'Assignment creation steps' and the 'Set up autograding and feedback' section. A yellow box highlights the configuration for a specific test case.

**Bước 5 – Grading and feedback**

- Add test
- run\_command
- Save test case

**5a** Add test

**5b** Choose an autograding method: input\_output, run\_command, java, node

**5c** Save test case

**Test name \***  
Hello project - Build Test

**Setup command**  
../setup.sh  
(Optional) Run before the test is run.

**Run command \***  
**gcc -o hello.c hello.exe** (highlighted)

**Timeout \***  
10  
The amount of minutes the test is allowed to run before it is killed. Maximum 60 minutes.

**Points**  
5  
(Optional) The amount of points awarded if the test passes.

Show grader repository

# Tạo Assignment Auto-grading



## Bước 5 –

### Grading and feedback

- d. Configuration auto grading
- e. Create assignment

**Set up autograding and feedback.**

**Assignment creation steps**

- ✓ Assignment basics
- ✓ Starter code and environment
- Grading and feedback**

**Add autograding tests**  
Autograding tests help provide feedback for students immediately upon submission using GitHub Actions. Add a test to enable autograding.

[GitHub Preset](#) [Custom YAML](#) [New](#) [Give feedback](#)

Hello project - Build Test [Edit](#) [Delete](#)

[+ Add test](#)

**Configure when autograding tests are run** [New](#) [Give feedback](#)

Every time a student submits an assignment  
 On a schedule (Timezone: Asia/Saigon)  
 Manually  
You will manually trigger autograding test runs for all students from the assignment dashboard.

**Create assignment**

5d

5e



# Share Assignment link

The screenshot shows a web browser window with the URL <https://classroom.github.com/classrooms/227212486-com108-udpm-classroom-class-su25-01/assignments/lab-hello-auto-grading>. The page title is "LAB-hello-auto-grading" and it states "Starter code from COM108-UDPM/com108-udpm-classroom-class-su25-01-lab-hello-auto-grading-FPOLY-Hello-auto-grading". The assignment is marked as "Active". The "Assignment Details" section shows the following statistics:

Students total	Accepted assignments	Assignment submissions
5 5 Rostered 0 Added students	0 0 Students	0 0 Submitted 0 Not submitted

Below the details are filter options: "Filters", "Search for an assignment", "Filter by unlinked accounts", "Filter by accepted", "Filter by passing", and "Sort". A large orange callout bubble with the number "6" is positioned over the "Assignment submissions" section. Another callout bubble with the number "6" is positioned over the "Assignment submissions" count in the first row.

**Bước 6 –**  
**Share assignment link**

Share the invitation link with your students so they can accept the assignment.

<https://classroom.git>

# Accept Assignment link

The screenshot shows a browser window for GitHub Classroom at the URL [classroom.github.com/assignment-invitations/ccee908bf2d14595f02fa91dfafafef7b](https://classroom.github.com/assignment-invitations/ccee908bf2d14595f02fa91dfafafef7b). The page displays the following content:

GitHub Classroom

COM108-UDPM-classroom-class-SU25-01

Accept the assignment —  
LAB-hello-auto-grading

Once you accept this assignment, you will be granted access to the [lab-hello-auto-grading-TranTuanAPC](#) repository in the [COM108-UDPM](#) organization on GitHub.

[Accept this assignment](#)

A yellow callout box with a black border and rounded corners is positioned on the right side of the page, pointing towards the "Accept this assignment" button. The text inside the callout box is:

**Sinh viên nhận link  
và Accept assignment**

# VS Code – Clone assignment & Do homework

The screenshot shows the Visual Studio Code (VS Code) interface. The title bar displays the path: lab-hello-auto-grading-TranTuanAPC. The left sidebar features the Source Control, Repositories, and Changes sections. In the Changes section, there is a commit message: "Nộp bài Hello, by TuanAPC". Below it is a "Commit" button. The main workspace shows a C file named "hello.c". The code is as follows:

```
#include <stdio.h>
// Viết một chương trình C
// để in ra các từ "Hello, World!"
// Ví dụ như sau
// main() {
//     // In ra "Hello, World!"
//     // Kết thúc chương trình
// }
int main() {
    // In ra "Hello, World!"
    printf("Hello, World!\n");
    // Kết thúc chương trình
}
```

The status bar at the bottom shows the file name "main\*", line "Ln 25, Col 1", and other settings like "Spaces: 4", "UTF-8", "CRLF", and "{} C".

# VS Code – Submit assignment

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it shows a repository named "lab-hello-auto-grading-TranTuanAPC". Under "REPOSITORIES", the "main" branch is selected. A message box is open, prompting the user to enter a commit message: "Message (Ctrl+Enter to commit on 'main')". Below the message box is a "Commit" button.
- Code Editor:** The main area displays a C program named "hello.c". The code is:

```
1 #include <stdio.h>
2
3 // Viết một chương trình C
4 // để in ra các từ "Hello, World!"
5
6 // Ví dụ như sau
7 // main() {
8
9 //     // In ra "Hello, World!"
10
11 //     // Kết thúc chương trình
12
13 // }
14
15 int main() {
16
17     // In ra "Hello, World!"
18
19     printf("Hello, World!\n");
20
21     // Kết thúc chương trình
22
23 }
24
25
```
- Graph View:** On the right side of the interface, there is a "GRAPH" view showing a timeline of commits. One commit is highlighted with a blue circle and labeled "Nộp bài Hello, by TuanAPC vnTranTuan". A red arrow points from this commit to the "main" branch in the File Explorer.
- Bottom Status Bar:** The status bar at the bottom shows the file path "main", line "Ln 25, Col 1", and encoding "UTF-8".

Sinh viên làm bài và nộp bài

# Assignment auto-grading reporter



**Giảng viên nhận thông tin  
Trạng thái và kết quả (điểm)  
của các bài đã nộp**

Ví dụ

Bài đã nộp

Submitted

Trạng thái

**Failed**

LAB-hello-auto-grading

Starter code from COM108-UDPM/com108-udpm-classroom-class-su25-01-lab-hello-auto-grading-FPOLY-Hello-auto-grading

Individual assignment Due Oct 31, 2025, 16:00 UTC Active

Sync assignments https://classroom.github.com/a/oZIVgz5y Run tests Edit Download

**Assignment Details**

Students total 5	Accepted assignments 1	Assignment submissions 1	Passed students 0
5 Rostered	0 Added students	1 Submitted	0 Not submitted
1 Students		0/1 Passed	

Filters Search for an assignment Filter by unlinked accounts Filter by accepted Filter by passing Sort

**Classroom roster**

TuanAPC	Submitted	@TranTuanAPC	Latest commit now	- 2 commits	0/5
---------	-----------	--------------	-------------------	-------------	-----

# Assignment auto-grading reporter



The screenshot shows a GitHub Actions run for a project named "Hello project - Build Test". The run failed due to an error in the "Autograde Reporter" step. The error message indicates that the code did not return an integer value. A yellow box highlights the error message and the command used to run the test: "gcc hello.c -o hello.exe". Another yellow box highlights the "Test runner summary" table, which shows a total score of 0 out of 5.

```

1  ► Run classroom-resources/autograde-grading-reporter@v1
7  S Processing: hello-project-build-test
8  X Hello project - Build Test
9  Test code:
10 gcc hello.c -o hello.exe
11
12 Total points for hello-project-build-test: 0.00/5
13
14 Test runner summary
15
16 | Test Runner Name | Test Score | Max Score |
17 |                |           |           |
18 | hello-project-bui... | 0          | 5          |
19 |                |           |           |
20 | Total:          | 0          | 5          |
21
22 🏆 Grand total tests passed: 0/1
23
24 Notice: Points 0/5
25 Notice: {"totalPoints":0,"maxPoints":5}

```



# VS Code – Submit assignment

SOURCE CONTROL

REPOSITORIES

lab-hello-auto-grading-TranT main ✓

CHANGES

Message (Ctrl+Enter to commit on "main")

✓ Commit

GRAPH

- Nộp bài Hello, update return, by TuanAPC vnTranTuan [main]
- Nộp bài Hello, by TuanAPC vnTranTuan
- Nộp bài Hello, by TuanAPC vnTranTuan
- add deadline github-classroom[bot]
- Setting up GitHub Classroom Feedback github-classroom[bot]
- Github Classroom Feedback github-classroom[bot]
- Github Classroom Autograding Workflow github-classroom[bot]
- Initial commit github-classroom[bot]
- Create GitHub Classroom Autograding Workflow github-clas...

File Edit Selection View ... ← → 🔍 lab-hello-auto-grading-TranTuanAPC ⚙️

hello.c

```

C hello.c > main()
1 #include <stdio.h>
2
3 // Viết một chương trình C
4 // để in ra các từ "Hello, World!"
5
6 // Ví dụ như sau
7 // main() {
8
9 //     // In ra "Hello, World!"
10
11 //     // Kết thúc chương trình
12 // }
13
14
15 int main() {
16
17     // In ra "Hello, World!"
18     printf("Hello, World!\n");
19
20     // Kết thúc chương trình
21     return 0;
22 }
23
24
25

```

vnTranTuan (now) 🔍 Ln 22, Col 14 Spaces: 4 UTF-8 CRLF { } C ⚙️ Go Live Win32 ✓ Continue (NE) 🔔

**Sinh viên làm lại bài và nộp lại bài**

Bổ sung thêm “**return 0;**”

# Assignment auto-grading reporter



**Giảng viên nhận thông tin  
ĐÃ CẬP NHẬT  
Trạng thái và kết quả (điểm)  
của các bài đã nộp**

Ví dụ

Bài đã nộp

Submitted

Trạng thái

Passed

The screenshot shows the GitHub Classroom interface for an assignment titled "LAB-hello-auto-grading". The assignment is a Starter code from COM108-UDPM/com108-udpm-classroom-class-su25-01-lab-hello-auto-grading-FPOLY-Hello-auto-grading. It is an Individual assignment due Oct 31, 2025, 16:00 UTC, and is currently Active.

**Assignment Details:**

- Students total: 5 (5 Rostered, 0 Added students)
- Accepted assignments: 1 (1 Students)
- Assignment submissions: 1 (1 Submitted, 0 Not submitted)
- Passed students: 1 (1/1 Passed)

**Classroom roster:**

Student	Status	Latest commit	Score
TuanAPC @TranTuanAPC	Submitted	Latest commit 3 minutes ago	5/5

A red arrow points to the "Submitted" status of the student TuanAPC, and a yellow arrow points to the "5/5" score.

# Assignment auto-grading reporter



Giảng viên nhận thông tin  
KẾT QUẢ CHI TIẾT  
Add test run\_command

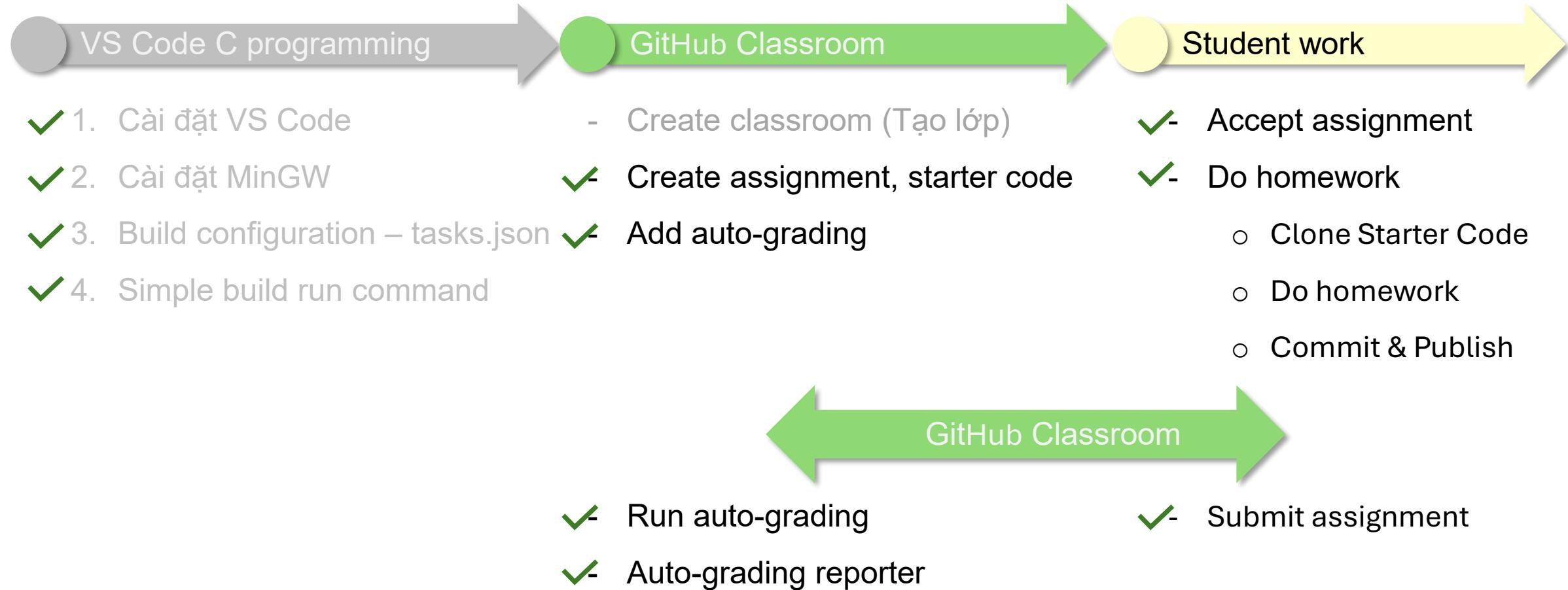
Summary  
Jobs  
**run-autograding-tests**  
Run details  
Usage  
Workflow file

run-autograding-tests  
succeeded 3 minutes ago in 7s  
Search logs

>Hello project - Build Test  
1 ▶ Run classroom-resources/autograde-command-grader@v1  
2 with:  
3 test-name: Hello project - Build Test  
4 command: gcc hello.c -o hello.exe  
5 timeout: 10  
6 max-score: 5

Autograding Reporter  
1 ▶ Run classroom-resources/autograde-grading-reporter@v1  
2 Processing: hello-project-build-test  
3 Hello project - Build Test  
4 Test code:  
5 gcc hello.c -o hello.exe  
6  
7 Total points for hello-project-build-test: 5.00/5  
8  
9 Test runner summary  
10  
11  
12 Test Runner Name | Test Score | Max Score  
13  
14 hello-project-bui... | 5 | 5  
15  
16 Total: | 5 | 5  
17  
18  
19  
20 Grand total tests passed: 1/1  
21  
22 Notice: Points 5/5  
23  
24 Notice: {"totalPoints":5,"maxPoints":5}

# Tóm tắt



# Tóm tắt



- Create classroom (Tạo lớp)
- ✓ Create assignment, starter code
- ✓ Add auto-grading
  
- ✓ ĐÃ THỰC HIỆN
  - Với trường hợp đơn giản, là
    - 1. Chương trình đơn giản  
một source file
    - 2. Lỗi đơn giản,  
lỗi biên dịch (**build error**)



## BÔ SUNG

- Với trường hợp phức tạp hơn, là
- 1. Chương trình **nhiều source files**
  - 2. **Tạo Test case** để kiểm tra,  
sử dụng thư viện **CUnit**



# VS Code – C Project multiple source files

The screenshot shows the VS Code interface for a C project with multiple source files.

**Explorer View:** Shows the project structure under 'SAMP...' (SampleL5B0). It includes a '.vscode' folder containing 'launch.json' and 'tasks.json', and source files 'main.c' and 'main\_old.c'. 'main\_old.c' is currently selected.

**Code Editor:** Displays the content of 'main\_old.c'.

```

1 #include <stdio.h>
2
3
4 int main() {
5     int so1, so2;
6     int Tong;
7
8     // Nhập vào 2 số từ người dùng
9     printf("Nhập vào số thứ nhất: ");
10    scanf("%d", &so1);
11
12    printf("Nhập vào số thứ hai: ");
13    scanf("%d", &so2);
14
15    Tong = so1 + so2;
16
17    // Hiển thị kết quả
18    printf("Tổng hai số là: %d\n", Tong);
19
20    return 0;
21 }

```

**Terminal:** Shows the output of the program execution.

```
Nhập vào số thứ nhất: 10
Nhập vào số thứ hai: 4
Tổng hai số là: 14
PS E:\WorkFolder\2025\AutoGrading\C\SampleL5B0>
```

**Tasks.json:** Shows the build configuration.

```

1 {
2     "version": "2.0.0",
3     "tasks": [
4         {
5             "label": "C/C++: gcc build SAMPLE",
6             "type": "shell",
7             "command": "gcc",
8             "args": [
9                 "-o",
10                "my_program",
11                "main.c",
12                "-g"
13            ],
14            "group": "build",
15            "detail": "Biên dịch toàn bộ SAM",
16            "problemMatcher": "$gcc"
17        }
18    ]
19 }

```

**C Project 1 source file**

**Và Build configuration**

# VS Code – C Project multiple source files

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows a file tree for a project named "SAMP...". It contains:
  - `.vscode` folder with `launch.json` and `tasks.json`.
  - `include` folder containing `my\_functions.h`.
  - `src` folder containing `my\_functions.c` and `main.c` (which is currently selected).
  - Other files: `main\_old.c`, `.gitignore`, and `my\_program.exe`.
- MAIN EDITOR:** Displays the content of `main.c`:
 

```
You, yesterday | 1 author (You)
1 #include <stdio.h>
2 #include "my_functions.h"
3
4 int main() {
5     int so1, so2;
6     int Tong;
7
8     printf("Nhập vào số thứ nhất: ");
9     scanf("%d", &so1);
10
11    printf("Nhập vào số thứ hai: ");
12    scanf("%d", &so2);
13
14    Tong = Tinh_Tong(so1, so2);
15
16    printf("Tổng hai số là: %d\n", Tong);
17
18    return 0;
19 }
20
```
- SECOND EDITOR:** Displays the content of `my\_functions.h`:
 

```
include > C my_functions.h > ...
You, yesterday | 1 author (You)
1 #ifndef MY_FUNCTIONS_H
2 #define MY_FUNCTIONS_H
3
4 // Khai báo hàm Tinh_Tong
5 int Tinh_Tong(int a, int b);
6
7 #endif /* MY_FUNCTIONS_H */
8
```
- THIRD EDITOR:** Displays the content of `my\_functions.c`:
 

```
src > C my_functions.c > ...
You, yesterday | 1 author (You)
1 #include "my_functions.h"
2
3 // Định nghĩa hàm Tinh_Tong
4 int Tinh_Tong(int a, int b) {
5     return a + b;
6
7 }
```
- TERMINAL:** Shows the output of the program execution:
 

```
Nhập vào số thứ nhất: 10
Nhập vào số thứ hai: 4
Tổng hai số là: 14
PS E:\WorkFolder\2025\AutoGrading\C\SampleL5B0>
```

**C Project multiple source files  
Và Build configuration**



# VS Code – C Project multiple source files

The screenshot shows the VS Code interface with three main files open:

- file 1**: `main.c` (top left)
- file 2**: `my_functions.h` (top right)
- file 3**: `my_functions.c` (bottom left)

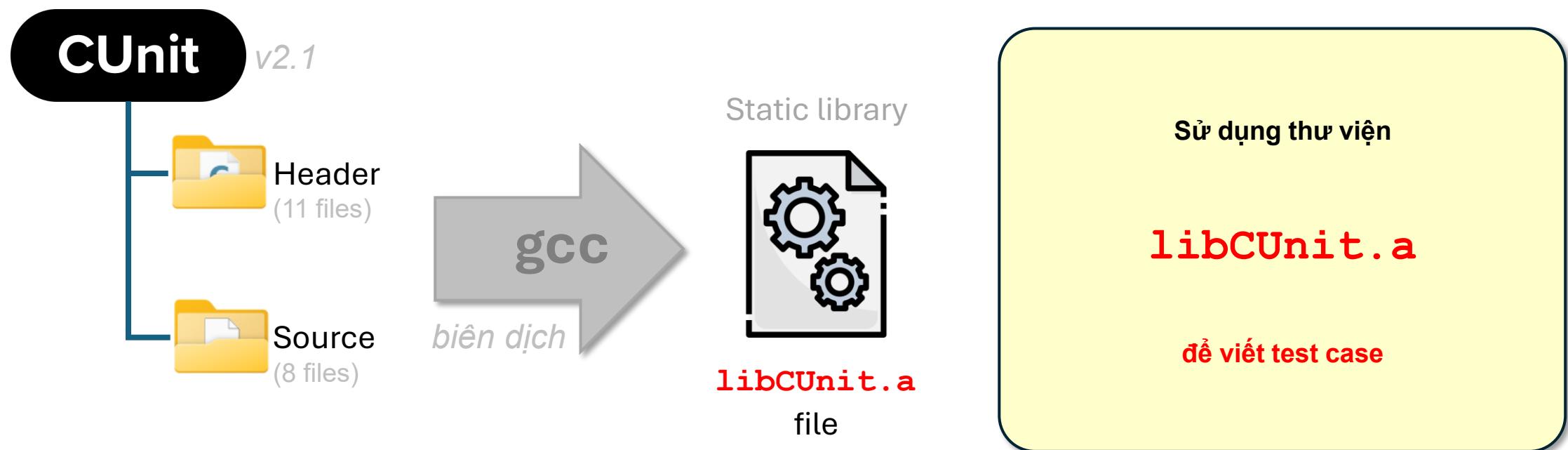
The `main.c` file contains C code for a program that adds two numbers and prints the result. The `my_functions.h` file defines a function `Tinh_Tong`. The `my_functions.c` file implements this function.

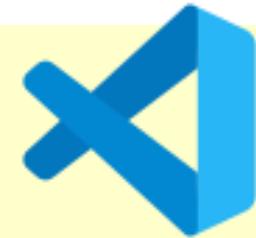
A yellow callout box at the bottom right contains the text:

**C Project multiple source files**  
**Và Build configuration**

## C Project **multiple source files** Và Build configuration

# GCC – build Cunit (local)





# VS Code – C Project & CUnit test case

**C Project multiple source files**  
**App & Test case**  
**Và multiple tasks configuration**

VS Code Explorer:

- SAMPLEC
- .vscode
- launch.json
- tasks.json
- include
- Automated.h
- Basic.h
- Console.h
- CUError.h
- CUnit\_intl.h
- CUnit.h
- my\_functions.h
- MyMem.h
- TestDB.h
- TestRun.h
- Util.h
- wxWidget.h
- lib
- libCUnit.a
- src
- my\_functions.c
- test
- test\_tinh\_tong.c
- main.c

File: test\_tinh\_tong.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "CUnit.h"
4 #include "Basic.h"
5 #include "my_functions.h"

6
7 void test_Tinh_Tong() {
8     // Test với các số nguyên
9     CU_ASSERT_EQUAL(Tinh_Tong(2, 3), 5);
10    CU_ASSERT_EQUAL(Tinh_Tong(0, 0), 0);
11    CU_ASSERT_EQUAL(Tinh_Tong(-1, 1), 0);
12    CU_ASSERT_EQUAL(Tinh_Tong(-2, -3), -5);

13
14    // Test với số thực
15    // sẽ bị lỗi vì hàm Tinh_Tong nhận int
16    CU_ASSERT_EQUAL(Tinh_Tong(1.5, 2.5), 4);

17 }

18
19 // Hàm khởi tạo và dọn dẹp test suite
20 int init_suite(void) { return 0; }
21 int clean_suite(void) { return 0; }

22
23 // Hàm chính để chạy unit test
24 > int main() { ... }
```

File: my\_functions.c

```
1 #include "my_functions.h"

2
3 // Định nghĩa hàm Tinh_Tong
4 int Tinh_Tong(int a, int b) {
5     return a + b;
6 }
```

File: tasks.json

```
1 {
2     "version": "2.0.0",
3     "tasks": [
4         // BUILD SAMPLE C
5         ...
6     ],
7
8     // BUILD TEST SAMPLE C
9     ...
10    ...
11
12    // RUN TEST
13    ...
14 }
```

task 1 build app

task 2 build test

task 3 run test

## C Project multiple source files

## App & Test case

### Và multiple tasks configuration



# VS Code – C Project & CUnit test case

## task 1 build app

```

4 // BUILD SAMPLE C
5 {
6     "label": "BUILD SAMPLE C",
7     "type": "shell",
8     "command": "gcc",
9     "args": [
10         "-o",
11         "my_program",
12         "main.c",
13         "src/my_functions.c",
14         "-I",
15         "include",
16         "-g"
17     ],
18     "group": "build",
19     "detail": "Biên dịch SAMPLE C",
20     "problemMatcher": "$gcc"
21 },
22

```

## task 2 build test

```

23 // BUILD TEST SAMPLE C
24 {
25     "label": "BUILD TEST SAMPLE C",
26     "type": "shell",
27     "command": "gcc",
28     "args": [
29         "-o",
30         "test_runner",
31         "test/test_tinh_tong.c",
32         "src/my_functions.c",
33         // Link CUnit Library, tại thư mục Lib
34         "-L", "lib", "-lCUnit",
35         "-I",
36         "include",
37         "-g"
38     ],
39     "group": "build",
40     "detail": "Biên dịch TEST SAMPLE C",
41     "problemMatcher": "$gcc"
42 },
43

```

## task 3 run test

```

44 // RUN TEST
45 {
46     "label": "run_test",
47     "type": "shell",
48     "command": "./test_runner",
49     "dependsOn": "BUILD TEST SAMPLE C",
50     "group": {
51         "kind": "test",
52         "isDefault": true
53     }
54 }
55

```

C Project multiple source files

App & Test case

Và multiple tasks configuration



# Tạo Auto-grading run\_command

## task 1 build app

```

4 // BUILD SAMPLE C
5 {
6     "label": "BUILD SAMPLE C",
7     "type": "shell",
8     "command": "gcc",
9     "args": [
10         "-o",
11         "my_program",
12         "main.c",
13         "src/my_functions.c",
14         "-I",
15         "include",
16         "-g"

```

## task 2 build test

```

23 // BUILD TEST SAMPLE C
24 {
25     "label": "BUILD TEST SAMPLE C",
26     "type": "shell",
27     "command": "gcc",
28     "args": [
29         "-o",
30         "test_runner",
31         "test/test_tinh_tong.c",
32         "src/my_functions.c",
33         // Link CUnit Library, tại thư mục Lib
34         "-L", "lib", "-lCUnit",
35         "-I",

```

**gcc -o test\_runner**  
**test/test\_tinh\_tong.c**  
**src/my\_functions.c**  
**-I include**  
**-Llib -lCUnit**

## task 3 run test

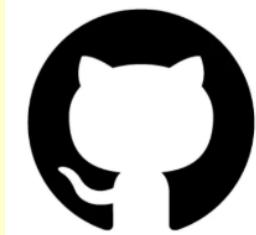
```

44 // RUN TEST
45 {
46     "label": "run_test",
47     "type": "shell",
48     "command": "./test_runner",
49     "dependsOn": "BUILD TEST SAMPLE C",
50     "group": {
51         "kind": "test",
52         "isDefault": true
53     }
54 }
55

```

**./test\_runner**

**gcc -o my\_program**  
**main.c**  
**src/my\_functions.c**  
**-I include**



# GitHub – C & CUnit Project, repository

The screenshot shows a GitHub repository page for 'vnTranTuan/sample-c-auto-grading'. The repository is described as a 'Public template'. It contains one branch ('main') and no tags. The repository was created by 'vnTranTuan' and has 1 commit. The commit message is 'Sample C, CUnit project, auto-grading'. The repository includes files for '.vscode', 'include', 'src', 'test', '.gitignore', and 'main.c'. A 'README' file is present, with a button to 'Add a README'. The repository has 0 stars, 0 forks, and 0 watching. A callout box highlights the repository as a 'C & CUnit Project Template repository'.

**C & CUnit Project  
Template repository**

# Tóm tắt



- Create classroom (Tạo lớp)
- ✓ Create assignment, starter code
- ✓ Add auto-grading
  
- ✓ ĐÃ THỰC HIỆN  
Với trường hợp đơn giản, là
  - ✓ 1. Chương trình đơn giản  
một source file
  - ✓ 2. Lỗi đơn giản,  
lỗi biên dịch (**build error**)



## BỒ SUNG

- Với trường hợp phức tạp hơn, là
- ✓ 1. Chương trình **nhiều source files**
  - ✓ 2. Tạo **Test case** để kiểm tra,  
sử dụng thư viện **CUnit**



# Tạo Assignment Auto-grading

Tiếp tục add auto-grading ở trang sau

Gồm các test như sau

**1. Test build app (5 điểm):**

để kiểm tra “build error”

**2. Build test (5 điểm):**

để biên dịch chương trình test

**3. Run test (5 điểm):**

để chạy chương trình test

**Tham khảo các bước Tạo assignment**

(slide phần 1)

a. Assignment basics

b. Starter code and environment

# Tạo Assignment Auto-grading, Test build app

The screenshot shows the GitHub Classroom interface for setting up auto-grading and feedback for a new assignment. A yellow box highlights the 'Test build app' configuration window.

**Test name \***: Test build app

**Setup command**: ./setup.sh  
(Optional) Run before the test is run.

**Run command \***: `gcc -o my_program main.c src/my_functions.c -I include`

**Timeout \***: 10  
The amount of minutes the test is allowed to run before it is killed. Maximum 60 minutes.

**Points**: 5  
(Optional) The amount of points awarded if the test passes.

**Show grader repository**

**Save test case**

**Bước 5 – Grading and feedback**

- a. Add test
- b. run\_command
- c. Save test case

5a      5b      5c

# Tạo Assignment Auto-grading, Build test

**Bước 5 – Grading and feedback**

- a. Add test
- b. run\_command
- c. Save test case

5a

5b

5c

**Test name \***  
Build test

**Setup command**  
.setup.sh  
(Optional) Run before the test is run.

**Run command \***

```
gcc -o test_runner test/test_tinh_tong.c
src/my_functions.c -l include -Llib -lCUnit
```

10      5  
The amount of minutes the test is allowed to run before it is killed. Maximum 60 minutes.      (Optional) The amount of points awarded if the test passes.

Show grader repository

Save test case



# Tạo Assignment Auto-grading, Build test

**Run\_command Build test** có thể sử dụng 2 cách sau :

**1** CUnit đã được biên dịch

- Lệnh ngắn
- CUnit đã được biên dịch tại máy local
- Trình biên dịch ở local và GitHub Server có thể khác nhau, và dẫn đến “error” khi chạy “auto-grading”

**2** CUnit SẼ được biên dịch ở GitHub Server

- Lệnh đầy đủ
- CUnit source files cần được upload lên GitHub Server
- CUnit và chương trình cùng được biên dịch ở GitHub Server, và phù hợp để chạy “auto-grading”

**1**

```
gcc -o test_runner test/test_tinh_tong.c
src/my_functions.c -I include -Llib -lCUnit
```

**2**

```
gcc -c cunit-src/*.c -I include && ar rcs
lib/libCUnit.a *.o && gcc -o test_runner
test/test_tinh_tong.c src/my_functions.c -I
include -Llib -lCUnit
```

# Tạo Assignment Auto-grading, Build test



**Run\_command** có trật tự và nội dung như sau :

- “**gcc**” : lệnh biên dịch
- **-o test\_runner** :  
kết quả là file “test\_runner.exe”
- **test/test\_tinh\_tong.c** :  
source của chương trình kiểm thử
- **src/my\_functions.c** :  
có sử dụng thư viện tự tạo
- **-I include** : thư mục chứa các header
- **-Llib** : chỉ định thư mục “lib” chứa file thư viện CUnit đã được biên dịch
- **-lCUnit** :  
liên kết thư viện CUnit vào chương trình

Test name \*

Setup command

(Optional) Run before the test is run.

Run command \*

```
gcc -o test_runner test/test_tinh_tong.c  
src/my_functions.c -I include -Llib -lCUnit
```

10      5

The amount of minutes the test is allowed to run before it is killed. Maximum 60 minutes.      (Optional) The amount of points awarded if the test passes.

Show grader repository

Save test case

1



# Tạo Assignment Auto-grading, Build test

**Run\_command** có trật tự và nội dung như sau :

- “`gcc -c cunit-src/*.c -I include`” :  
lệnh biên dịch CUnit source files thành objects
- `ar rcs lib/libCUnit.a *.o` :  
dịch các objects và tạo thành thư viện tĩnh  
“`libCUnit.a`”
- `&&` :  
nối các thành phần lệnh và đảm bảo thực hiện  
thành công thành phần lệnh trước rồi đến thành  
phần lệnh sau của toàn bộ câu lệnh
- Thành phần lệnh còn lại tương tự như cách ①

Test name \*

Setup command

(Optional) Run before the test is run.

Run command \*

```
gcc -c cunit-src/*.c -I include && ar rcs
lib/libCUnit.a *.o && gcc -o test_runner
test/test_tinh_tong.c src/my_functions.c -I
include -Llib -lCUnit
```

Show grader repository

Save test case

2

# Tạo Assignment Auto-grading, Run test

**Bước 5 – Grading and feedback**

- Add test
- run\_command
- Save test case

5a

5b

5c

**Test name \***  
Run test

**Setup command**  
.setup.sh  
(Optional) Run before the test is run.

**Run command \***  
.test\_runner

**Timeout \***  
10  
The amount of minutes the test is allowed to run before it is killed. Maximum 60 minutes.

**Points**  
5  
(Optional) The amount of points awarded if the test passes.

Show grader repository

Save test case

# Tạo Assignment Auto-grading, các tests



## Assignment creation steps

- ✓ Assignment basics
- ✓ Starter code and environment
- Grading and feedback**

Add autograding tests  
Autograding tests help provide feedback for students immediately upon submission using [GitHub Actions](#). Add a test to enable autograding.

[GitHub Preset](#) [Custom YAML](#) [New](#) [Give feedback](#)

Test build app	<a href="#"></a> <a href="#"></a>
Build test	<a href="#"></a> <a href="#"></a>
Run test	<a href="#"></a> <a href="#"></a>

[+ Add test ▾](#)

Configure when autograding tests are run [New](#) [Give feedback](#)

Every time a student submits an assignment  
 On a schedule (Timezone: Asia/Saigon)  
 Manually

You will manually trigger autograding test runs for all students from

**Bước 5 –**  
**Grading and feedback**

- a. Add test
- b. run\_command
- c. Save test case

Assignment đã được tạo 3 test

- 1. Test build app**
- 2. Build test**
- 3. Run test**

# Assignment Auto-grading

The screenshot shows a GitHub Classroom interface for an assignment titled "ASM sample C auto-grading". The assignment is set as an individual assignment due on Oct 31, 2025, at 16:00 UTC. It is currently active. The assignment has been submitted by two students, both of whom have passed. The "Assignment Details" section displays the following statistics:

- Students total: 5 (5 Rostered, 0 Added students)
- Accepted assignments: 2 (2 Students)
- Assignment submissions: 2 (2 Submitted, 0 Not submitted)
- Passed students: 2/2 Passed

Below this, the "Classroom roster" lists two students:

- TuanAPC**: Submitted, latest commit 11 minutes ago, 6 commits, 15/15.
- TuanTA71**: Submitted, latest commit 6 minutes ago, 1 commit, 15/15.

At the bottom, there is a "View commits" button.

Assignment auto-grading đã được áp dụng, với mỗi bài nộp. Các thông tin như

1. Passed hay Failed
2. Điểm số
3. Chi tiết repository đã nộp
4. ...

# Auto-grading reporter, Detail report

The screenshot shows a GitHub Actions job named "run-autograding-tests" for a repository "COM108-UDPM / asm-sample-c-auto". The job status is "succeeded 5 minutes ago in 17s". The workflow steps include "Set up job", "Checkout code", "Test build app", "Build test", and "Run test". The "Run test" step is expanded, showing the command "Run classroom-resources/autograding-command-grader@v1" and the CUnit output. A red box highlights the test results, which show a failed assertion: "Test: Tinh\_Tong Test 1 ...FAILED" and "1. test/test\_tinh\_tong.c:16 - CU\_ASSERT\_EQUAL(Tinh\_Tong(1.7, 2.9),4.6)". A yellow box highlights the entire CUnit output. The "Run Summary" table shows 1 suite, 1 test, and 5 asserts, with 1 failed assert.

Run Summary:	Type	Total	Ran	Passed	Failed	Inactive
	suites	1	1	n/a	0	0
	tests	1	1	0	1	0
	asserts	5	5	4	1	n/a

Kết quả các test case đã viết trong chương trình kiểm thử

# Tóm tắt



## GitHub Classroom

- Create classroom (Tạo lớp)
- ✓ Create assignment, starter code
- ✓ Add auto-grading



## C & CUnit project

### ĐÃ THỰC HIỆN

Với trường hợp đơn giản, là

- ✓ 1. Chương trình đơn giản

một source file

- ✓ 2. Lỗi đơn giản,

lỗi biên dịch (**build error**)



### BỒ SUNG

Với trường hợp phức tạp hơn, là

- ✓ 1. Chương trình **nhiều source files**

- ✓ 2. Tạo **Test case** để kiểm tra,

sử dụng thư viện **CUnit**

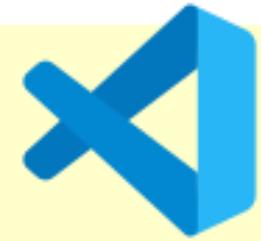


## Auto-grading

### ĐÃ THỰC HIỆN

1. Add test, run command

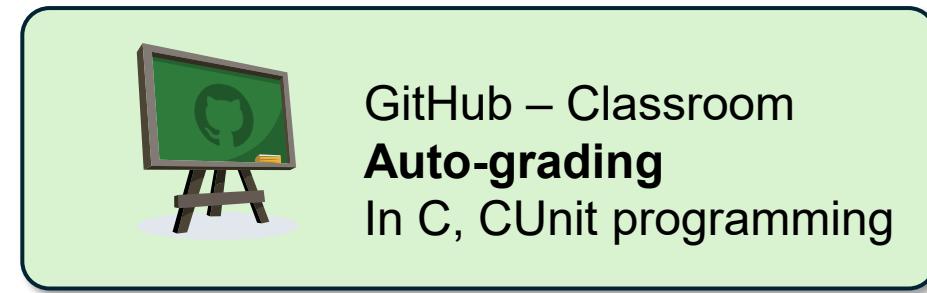
2. Auto-grading reporter



# VS Code – launch.json (debugger configuration)

Đang thực hiện ...

# HẾT PHẦN 2 – Nhóm 1



# TRÂN TRỌNG CẢM ƠN

09-2025  
Nhóm thực hiện  
Thân Hoàng Lộc  
Trần Quang Bình  
Trần Văn Huy  
Trần Anh Tuấn

# Tham khảo

1. <https://cunit.sourceforge.net/>
2. <https://cmake.org/>