**📘 README.md — Taller de Pruebas de Software en C++ con Catch2**

**🧪 Objetivo del taller**

Este taller tiene como propósito aplicar pruebas unitarias sobre un conjunto de clases en C++: Docente, Estudiante y Notas. Se utiliza la librería **Catch2** para realizar las pruebas en un entorno de desarrollo moderno con **Visual Studio Code** y **vcpkg** como gestor de dependencias.

**📁 Estructura del proyecto**

pgsql

CopiarEditar

taller\_pruebas\_cpp/

├── clases/

│ ├── Docente.h

│ ├── Estudiante.h

│ ├── Notas.h

│ └── implementaciones.cpp

├── tests/

│ └── pruebas.cpp

├── .vscode/

│ ├── tasks.json

│ └── launch.json

└── README.md

**🔧 Requisitos previos**

* Tener **VSCode** instalado
* Tener instalado el compilador g++ (MinGW o MSYS2)
* Tener gdb.exe disponible para depuración
* Tener vcpkg clonado y configurado

**⚙️ Instalación de dependencias**

1. Clona o descarga el repositorio de vcpkg:

bash

git clone https://github.com/microsoft/vcpkg

cd vcpkg

.\bootstrap-vcpkg.bat

1. Instala **Catch2** con arquitectura personalizada:

bash

.\vcpkg.exe install catch2:x64-mingw-custom

1. (Opcional) Integra vcpkg con VSCode:

bash

.\vcpkg integrate install

**🛠 Compilar y correr pruebas (desde VSCode)**

1. Abre el proyecto en Visual Studio Code
2. Presiona Ctrl+Shift+B para **compilar** (tasks.json)
3. Presiona F5 para **ejecutar con depuración** (launch.json)

**📌 Descripción de Clases**

* Docente: Almacena nombre y cédula del profesor
* Estudiante: Contiene nombre y código del estudiante
* Notas: Guarda tres notas y permite calcular el promedio

**✅ Pruebas Incluidas**

Usando **Catch2**, se validan:

* Correcta inicialización de datos en Docente y Estudiante
* Precisión del cálculo de promedio en Notas

**🧠 Debes saber:**

Este proyecto en clase es un ejemplo práctico para:

* Introducir pruebas unitarias en C++
* Practicar el uso de gestores de dependencias modernos
* Practicar integración entre código fuente y pruebas automatizadas

# Guía visual paso a paso para compilar y ejecutar tu proyecto de pruebas en C++ con Catch2, vcpkg y Visual Studio Code en Windows.

Esta guía está pensada para estudiantes o docentes que trabajan en entornos educativos o de práctica.

**✅ Paso 1: Instalar herramientas necesarias**

**🧰 Herramientas requeridas:**

| **Herramienta** | **Descripción** | **Enlace** |
| --- | --- | --- |
| **VSCode** | Editor de código | <https://code.visualstudio.com/> |
| **g++ / MinGW** | Compilador para C++ | <https://www.mingw-w64.org/> |
| **vcpkg** | Gestor de librerías C++ | <https://github.com/microsoft/vcpkg> |
| **GDB** | Depurador (incluido en MinGW) |  |

Verifica que g++ y gdb estén en tu PATH ejecutando en la terminal:

bash

g++ --version

gdb --version

**✅ Paso 2: Clonar e instalar vcpkg**

1. Abre PowerShell o CMD.
2. Clona el repositorio:

bash

git clone https://github.com/microsoft/vcpkg

cd vcpkg

.\bootstrap-vcpkg.bat

1. Instala **Catch2** (usa la arquitectura adecuada según MinGW):

bash

.\vcpkg.exe install catch2:x64-mingw-custom

1. Integra vcpkg con VSCode (opcional pero recomendado):

bash

.\vcpkg.exe integrate install

**✅ Paso 3: Crear la estructura del proyecto**

Dentro de una carpeta llamada taller\_pruebas\_cpp, organiza los archivos así:

java

CopiarEditar

taller\_pruebas\_cpp/

├── clases/ ← contiene las clases (.h, .cpp)

├── tests/ ← contiene pruebas Catch2

├── .vscode/ ← configuraciones de tareas

Copia aquí los archivos:

* Docente.h, Estudiante.h, Notas.h, implementaciones.cpp
* pruebas.cpp con las pruebas unitarias
* tasks.json, launch.json

**✅ Paso 4: Configurar .vscode/tasks.json**

Este archivo compila tus pruebas:

json

CopiarEditar

{

"version": "2.0.0",

"tasks": [

{

"label": "Compilar pruebas",

"type": "shell",

"command": "g++",

"args": [

"clases/implementaciones.cpp",

"tests/pruebas.cpp",

"-std=c++17",

"-I${workspaceFolder}/vcpkg/installed/x64-mingw-custom/include",

"-L${workspaceFolder}/vcpkg/installed/x64-mingw-custom/lib",

"-o", "pruebas.exe"

],

"group": {

"kind": "build",

"isDefault": true

},

"problemMatcher": ["$gcc"]

}

]

}

**✅ Paso 5: Configurar .vscode/launch.json**

Este archivo lanza y depura el ejecutable:

json

{

"version": "0.2.0",

"configurations": [

{

"name": "Ejecutar pruebas.exe",

"type": "cppdbg",

"request": "launch",

"program": "${workspaceFolder}/pruebas.exe",

"stopAtEntry": false,

"cwd": "${workspaceFolder}",

"externalConsole": true,

"MIMode": "gdb",

"miDebuggerPath": "gdb.exe",

"setupCommands": [

{

"description": "Enable pretty-printing for gdb",

"text": "-enable-pretty-printing",

"ignoreFailures": true

}

]

}

]

}

**✅ Paso 6: Ejecutar desde VSCode**

1. Abre la carpeta del proyecto desde **Visual Studio Code**
2. Compila el proyecto:
   * Ctrl + Shift + B (selecciona **Compilar pruebas** si te lo pregunta)
3. Ejecuta el programa:
   * Presiona F5 (depura con GDB y abre una consola externa)

**🧪 Resultado esperado**

* Una ventana de consola muestra los resultados de Catch2, por ejemplo:

bash

===============================================================================

All tests passed (3 assertions in 3 test cases)

**🧠 Recomendaciones**

* Puedes agregar más pruebas en tests/pruebas.cpp
* Usa Approx(valor).epsilon(0.01f) para comparar decimales
* Activa la terminal integrada para ver errores de compilación cómodamente