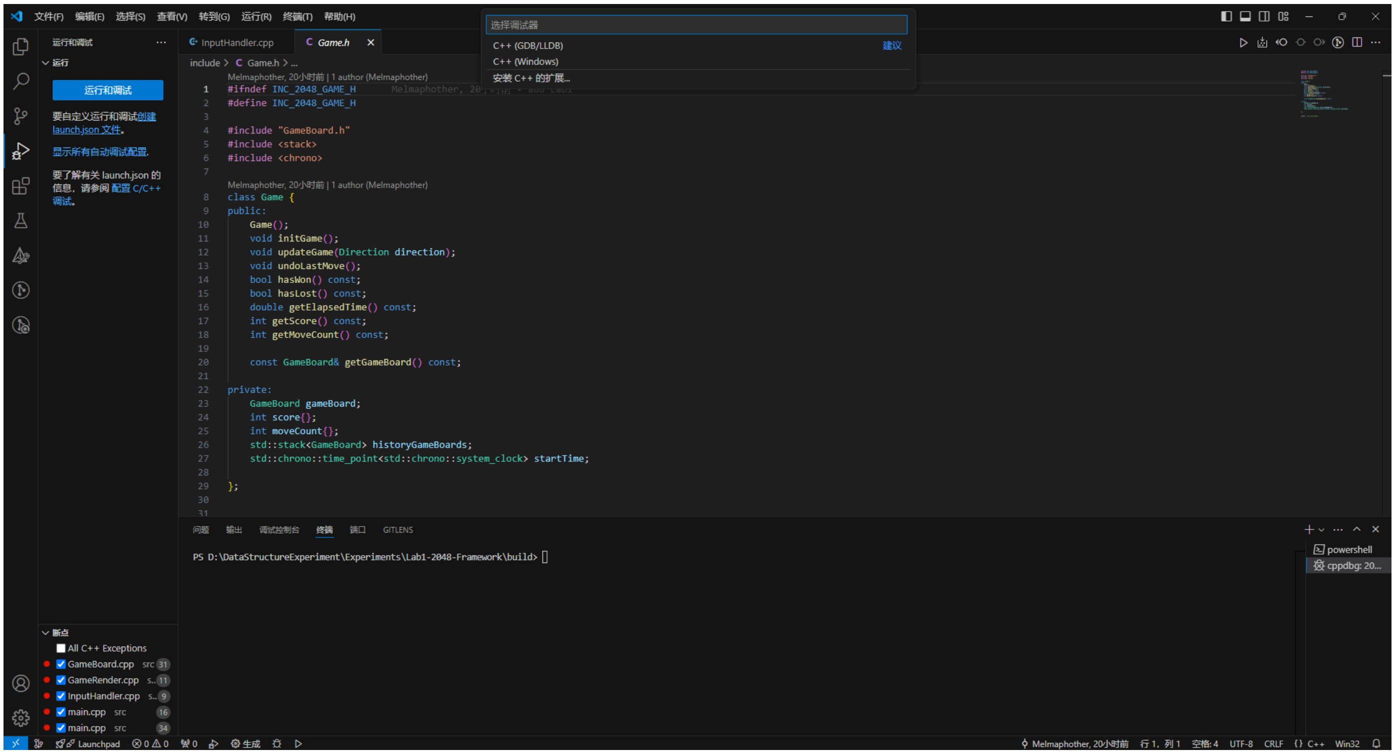
项目调试步骤说明

项目调试之前需要对vscode的调试配置进行相关设置。

- 首先找到左边一栏带小虫子的播放键图标，点击后进入 `创建launch.json文件` ：

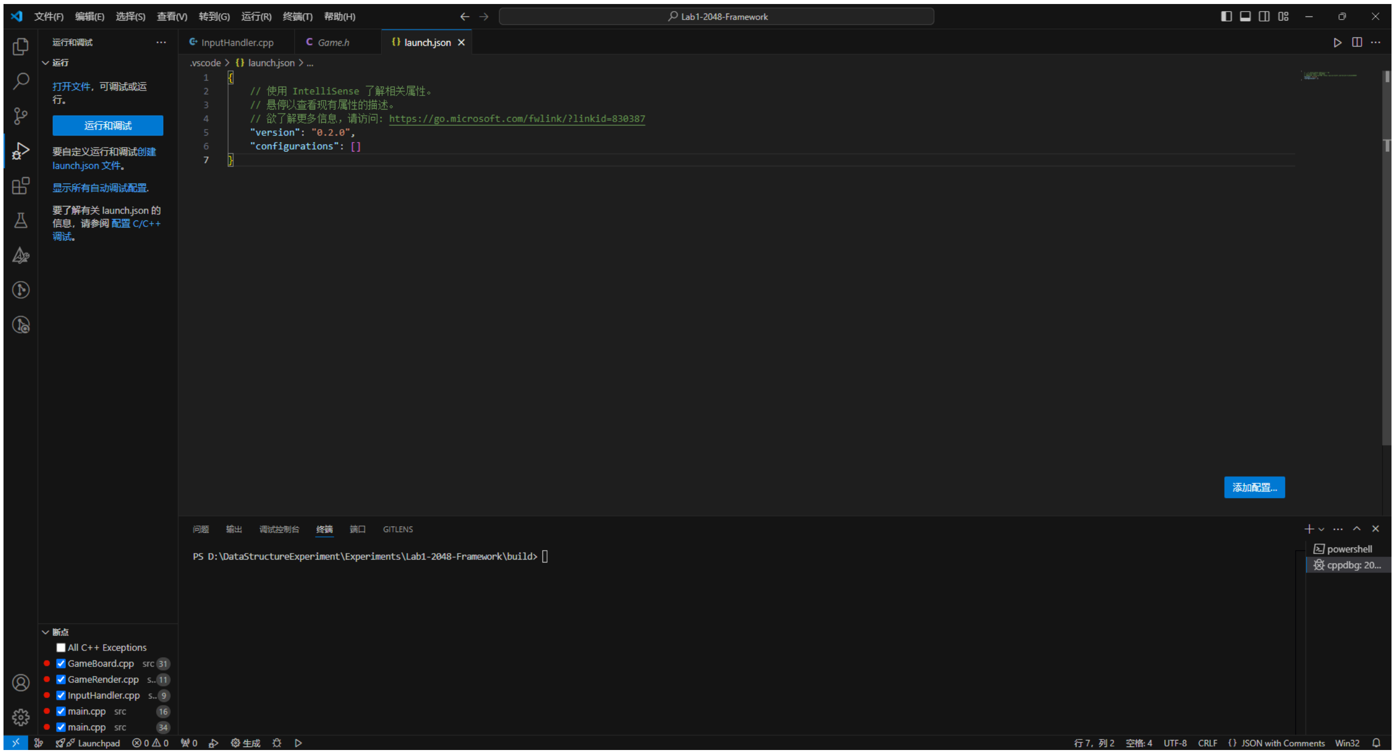


选择调试器
C++ (GDB/LLDB)
建议
C++ (Windows)
安装 C++ 的扩展...
运行和调试
要自定义运行和调试创建 launch.json 文件。
显示所有自动调试配置。
要了解有关 launch.json 的信息，请参阅 配置 C/C++ 调试。

```cpp
#ifndef INC_2048_GAME_H
#define INC_2048_GAME_H

#include "GameBoard.h"
#include <stack>
#include <chrono>

class Game {
public:
    Game();
    void initGame();
    void updateGame(Direction direction);
    void undoLastMove();
    bool hasWon() const;
    bool hasLost() const;
    double getElapsedTime() const;
    int getScore() const;
    int getMoveCount() const;

    const GameBoard& getGameBoard() const;

private:
    GameBoard gameBoard;
    int score{};
    int moveCount{};
    std::stack<GameBoard> historyGameBoards;
    std::chrono::time_point<std::chrono::system_clock> startTime;

};
```

```
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build>
```

- 选择 `C++ (GDB/ LLDB)` ，进入如下界面：



运行和调试
打开文件，可调试或运行。
要自定义运行和调试创建 launch.json 文件。
显示所有自动调试配置。
要了解有关 launch.json 的信息，请参阅 配置 C/C++ 调试。
添加配置...

```json
{
    // 使用 IntelliSense 了解相关属性。
    // 悬停以查看现有属性的描述。
    // 欲了解更多信息，请访问: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": []
}
```

```
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build>
```

- 将 `launch.json` 文件里其中的配置内容修改为：

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Debug 2048",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/build/2048.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath":
            "D:\\download\\x86_64-8.1.0-release-posix-sjlj-rt_v6-rev0\\mingw64\\bin\\gdb.exe",
             // 修改成实际的gdb路径！注意\bin会被识别成\b in，记得加多一个反斜杠
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ],
            "preLaunchTask": "build",
            "internalConsoleOptions": "openOnSessionStart",
            "logging": {
                "engineLogging": true
            },
            "visualizerFile": "${workspaceFolder}/.vscode/gdb_visualizers.visualizers"
        }
    ]
}
```

它会将 `build/2048.exe` 作为需要进行调试运行的程序，
注意: 复制后需要修改 `"miDebuggerPath"` 为实际的gdb路径！

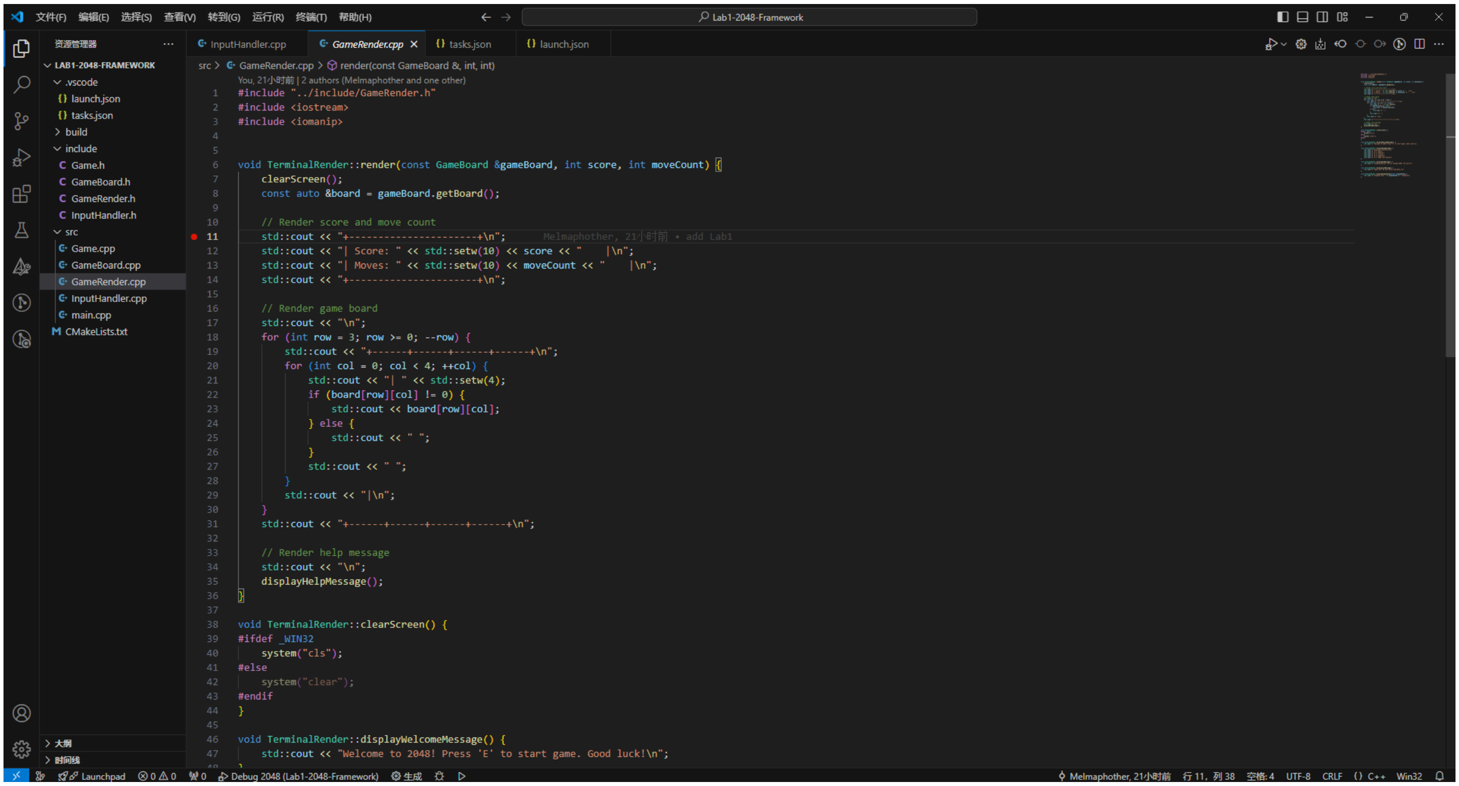
- 接着在 `.vscode` 目录下创建一个 `tasks.json` ，内容如下：

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "create_build_directory",
            "type": "shell",
            "command": "powershell",
            "args": [
                "-Command",
                "if (!(Test-Path -Path \"${workspaceFolder}/build\")) { New-Item -ItemType Dire
            ],
            "problemMatcher": [],
            "options": {
                "cwd": "${workspaceFolder}"
            }
        },
        {
            "label": "configure",
            "type": "shell",
            "command": "cmake",
            "args": [
                "-G", "MinGW Makefiles",
                "-DCMAKE_BUILD_TYPE=Debug",
                ".."
            ],
            "options": {
                "cwd": "${workspaceFolder}/build"
            },
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "dependsOn": "create_build_directory",
            "problemMatcher": [],
            "detail": "Generated task by CMake."
        },
        {
            "label": "build",
            "type": "shell",
            "command": "mingw32-make",
            "args": [],
            "options": {
                "cwd": "${workspaceFolder}/build"
            },
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "dependsOn": "configure",
            "problemMatcher": ["$gcc"],
            "detail": "Generated task by CMake."
        }
    ]
}
```

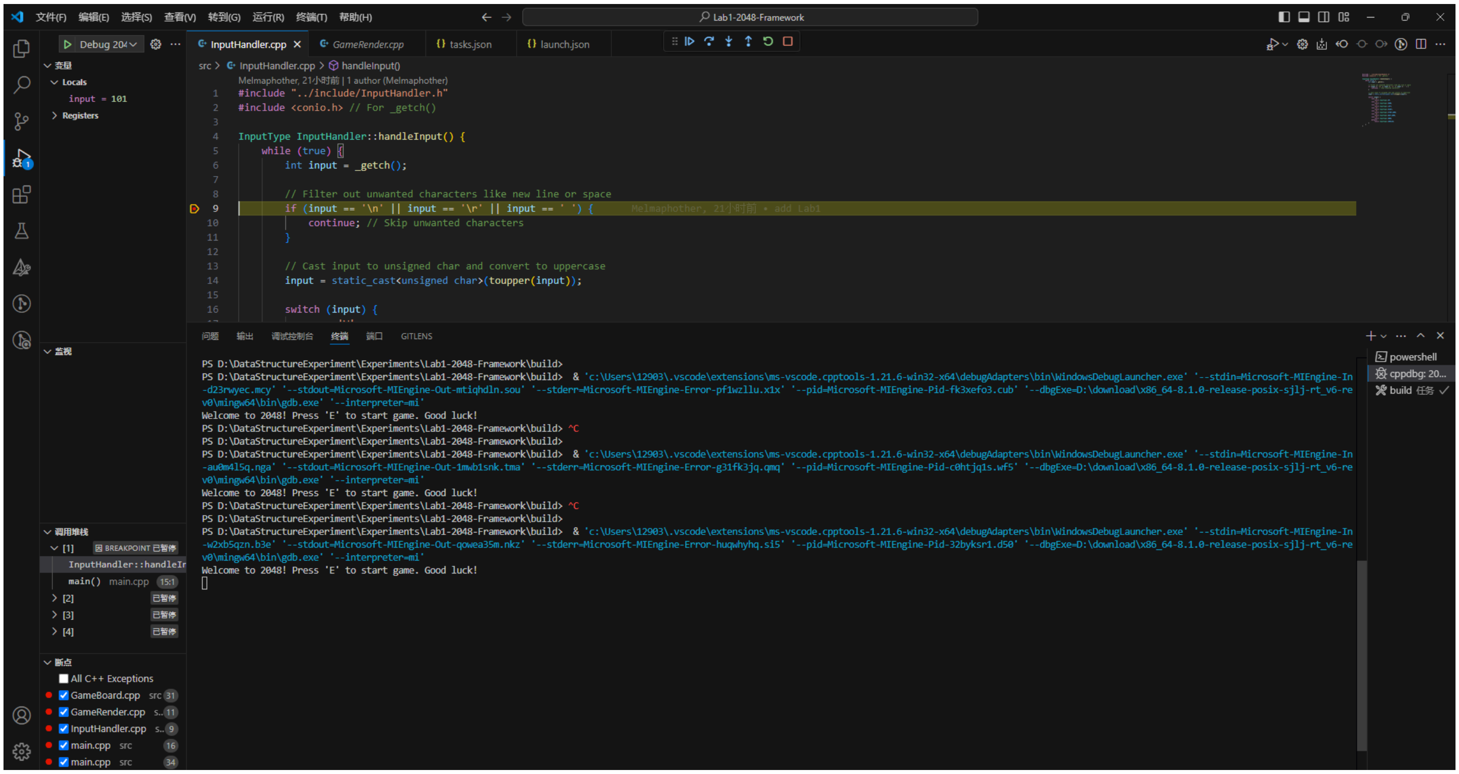
其中的内容包括：

- 检查 `build` 目录是否存在，如果不存在则新建一个目录，用于存放待会儿脚本生成的内容；
- 进行 `makefile` 的创建，添加了调试参数 `-DCMAKE_BUILD_TYPE=Debug` ；
- 进行项目的构建。

在配置完这些内容后，在需要调试的代码行前打上调试断点（图中红点），按下F5就能进行调试，看到中间的调用情况和局部变量的值了。



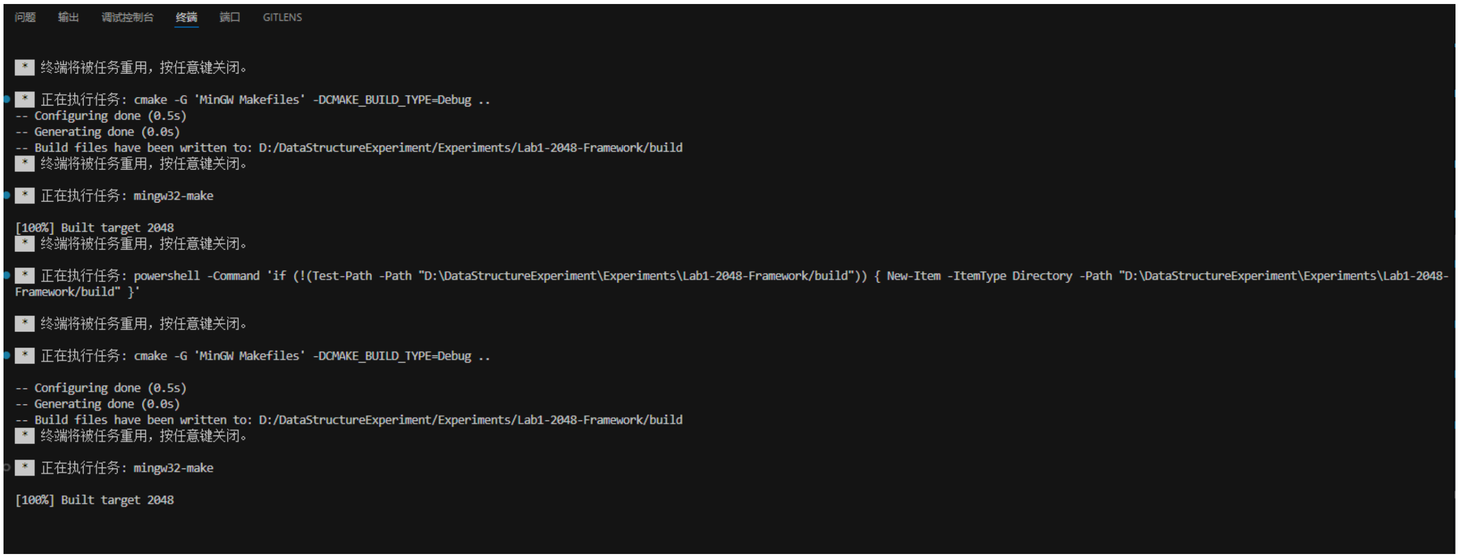
```cpp
#include "../include/GameRender.h"
#include <iostream>
#include <iomanip>


void TerminalRender::render(const GameBoard &gameBoard, int score, int moveCount) {
    clearScreen();
    const auto &board = gameBoard.getBoard();

    // Render score and move count
    std::cout << "+---------------------+\n";
    std::cout << "| Score: " << std::setw(10) << score << "    |\n";
    std::cout << "| Moves: " << std::setw(10) << moveCount << "    |\n";
    std::cout << "+---------------------+\n";

    // Render game board
    std::cout << "\n";
    for (int row = 3; row >= 0; --row) {
        std::cout << "+------+------+------+------+\n";
        for (int col = 0; col < 4; ++col) {
            std::cout << "| " << std::setw(4);
            if (board[row][col] != 0) {
                std::cout << board[row][col];
            } else {
                std::cout << " ";
            }
            std::cout << " ";
        }
        std::cout << "|\n";
    }
    std::cout << "+------+------+------+------+\n";

    // Render help message
    std::cout << "\n";
    displayHelpMessage();
}

void TerminalRender::clearScreen() {
#ifdef _WIN32
    system("cls");
#else
    system("clear");
#endif
}

void TerminalRender::displayWelcomeMessage() {
    std::cout << "Welcome to 2048! Press 'E' to start game. Good luck!\n";
```



```cpp
#include "../include/InputHandler.h"
#include <conio.h> // For _getch()

InputType InputHandler::handleInput() {
    while (true) {
        int input = _getch();

        // Filter out unwanted characters like new line or space
        if (input == '\n' || input == '\r' || input == ' ') {
            continue; // Skip unwanted characters
        }

        // Cast input to unsigned char and convert to uppercase
        input = static_cast<unsigned char>(toupper(input));

        switch (input) {
```

```
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build>
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build>  & 'c:\Users\12903\.vscode\extensions\ms-vscode.cpptools-1.21.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-d23rwyec.mcy' '--stdout=Microsoft-MIEngine-Out-mtiqhdln.sou' '--stderr=Microsoft-MIEngine-Error-pf1wzllu.x1x' '--pid=Microsoft-MIEngine-Pid-fk3xefo3.cub' '--dbgExe=D:\download\x86_64-8.1.0-release-posix-sjlj-rt_v6-rev0\mingw64\bin\gdb.exe' '--interpreter=mi'
Welcome to 2048! Press 'E' to start game. Good luck!
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build> ^C
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build>
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build>  & 'c:\Users\12903\.vscode\extensions\ms-vscode.cpptools-1.21.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-au0m4l5q.nga' '--stdout=Microsoft-MIEngine-Out-1mwb1snk.tma' '--stderr=Microsoft-MIEngine-Error-g31fk3jq.qmq' '--pid=Microsoft-MIEngine-Pid-c0htjq1s.wf5' '--dbgExe=D:\download\x86_64-8.1.0-release-posix-sjlj-rt_v6-rev0\mingw64\bin\gdb.exe' '--interpreter=mi'
Welcome to 2048! Press 'E' to start game. Good luck!
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build> ^C
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build>
PS D:\DataStructureExperiment\Experiments\Lab1-2048-Framework\build>  & 'c:\Users\12903\.vscode\extensions\ms-vscode.cpptools-1.21.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-w2xb5qzn.b3e' '--stdout=Microsoft-MIEngine-Out-qowea35m.nkz' '--stderr=Microsoft-MIEngine-Error-huqwhyhq.si5' '--pid=Microsoft-MIEngine-Pid-32byksr1.d50' '--dbgExe=D:\download\x86_64-8.1.0-release-posix-sjlj-rt_v6-rev0\mingw64\bin\gdb.exe' '--interpreter=mi'
Welcome to 2048! Press 'E' to start game. Good luck!
```

在执行脚本构建时会产生以下参考输出：



```
 *  终端将被任务重用，按任意键关闭。

 *  正在执行任务: cmake -G 'MinGW Makefiles' -DCMAKE_BUILD_TYPE=Debug ..
-- Configuring done (0.5s)
-- Generating done (0.0s)
-- Build files have been written to: D:/DataStructureExperiment/Experiments/Lab1-2048-Framework/build
 *  终端将被任务重用，按任意键关闭。

 *  正在执行任务: mingw32-make

[100%] Built target 2048
 *  终端将被任务重用，按任意键关闭。

 *  正在执行任务: powershell -Command 'if (!(Test-Path -Path "D:\DataStructureExperiment\Experiments\Lab1-2048-Framework/build")) { New-Item -ItemType Directory -Path "D:\DataStructureExperiment\Experiments\Lab1-2048-Framework/build" }'

 *  终端将被任务重用，按任意键关闭。

 *  正在执行任务: cmake -G 'MinGW Makefiles' -DCMAKE_BUILD_TYPE=Debug ..

-- Configuring done (0.5s)
-- Generating done (0.0s)
-- Build files have been written to: D:/DataStructureExperiment/Experiments/Lab1-2048-Framework/build
 *  终端将被任务重用，按任意键关闭。

 *  正在执行任务: mingw32-make

[100%] Built target 2048
```