Understanding tasks.json

The first time you run your program, the C++ extension creates a `tasks.json` file, which you'll find in your project's `.vscode` folder. `tasks.json` stores your build configurations.

https://code.visualstudio.com/docs/cpp/config-mingw

Your new `tasks.json` file should look similar to the JSON below:

```json
{
  "tasks": [
    {
      "type": "cppbuild",
      "label": "C/C++: g++.exe build active file",
      "command": "C:\\mingw64\\bin\\g++.exe",
      "args": [
        "-fdiagnostics-color=always",
        "-g",
        "${file}",
        "-o",
        "${fileDirname}\\${fileBasenameNoExtension}.exe"
      ],
      "options": {
        "cwd": "${fileDirname}"
      },
      "problemMatcher": ["$gcc"],
      "group": {
        "kind": "build",
        "isDefault": true
      },
      "detail": "Task generated by Debugger."
    }
  ],
  "version": "2.0.0"
}
```

https://code.visualstudio.com/docs/cpp/config-mingw

Note: You can learn more about `tasks.json` variables in from the following link:

https://code.visualstudio.com/docs/editor/variables-reference

The `command` setting specifies the program to run; in this case that is `g++`.

The `args` array specifies the command-line arguments passed to g++. These arguments are listed in this file in the specific order expected by the compiler.

This task tells g++ to take the active file (`${file}`), compile it, and create an output file (`-o` switch) in the current directory (`${fileDirname}`) with the same name as the active file but with the `.exe` extension (`${fileBasenameNoExtension}.exe`). For us, this results in `helloworld.exe`.

The `label` value is what you will see in the tasks list; you can name this whatever you like.

The `detail` value is what you will see as the description of the task in the tasks list. It's highly recommended to rename this value to differentiate it from similar tasks.

The `problemMatcher` value selects the output parser to use for finding errors and warnings in the compiler output. For GCC, you'll get the best results if you use the `$gcc` problem matcher.

From now on, the play button will read from `tasks.json` to figure out how to build and run your program. You can define multiple build tasks in `tasks.json`, and whichever task is marked as the default will be used by the play button. In case you need to change the default compiler, you can run **Tasks: Configure Default Build Task** in the Command Palette. Alternatively you can modify the `tasks.json` file and remove the default by replacing this segment:

```
    "group": {

        "kind": "build",

        "isDefault": true

    },
```

with this:

```
    "group": "build",
```

Modifying tasks.json

You can modify your `tasks.json` to build multiple C++ files by using an argument like `"${workspaceFolder}/*.cpp"` instead of `"${file}"`.This will build all `.cpp` files in your current folder. You can also modify the output filename by replacing `"${fileDirname}\\${fileBasenameNoExtension}.exe"` with a hard-coded filename (for example `"${workspaceFolder}\\myProgram.exe"`).