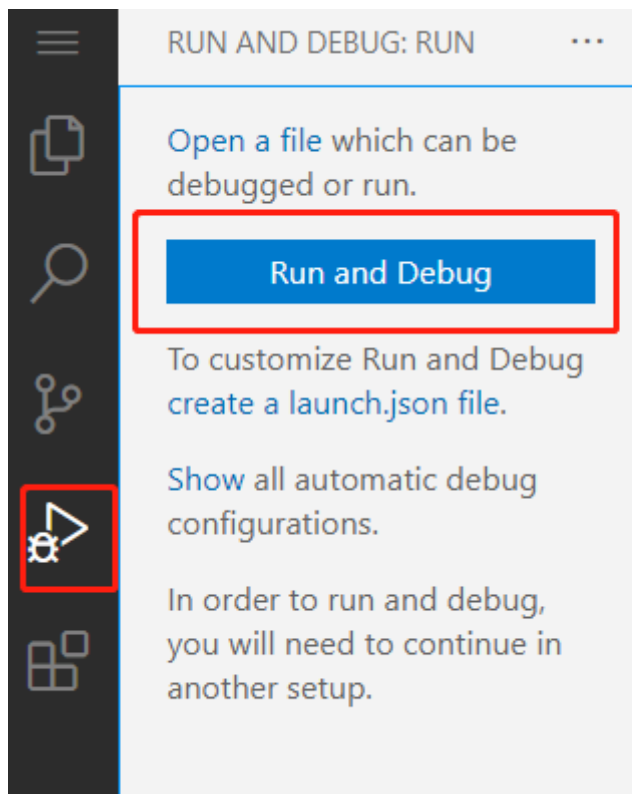


以 水杉 vscode 为例, 讲一下如何调试 c++ 代码.

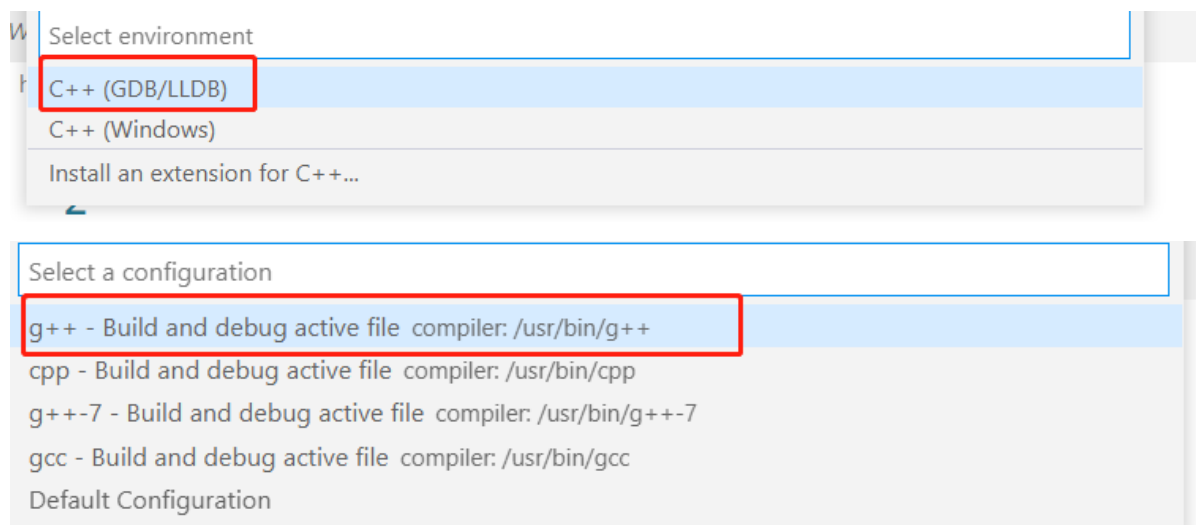
水杉 vscode 与自己本地安装的vscode 基本一致, 如果自己安装的vscode或编译器环境乱或者使用其他 IDE不熟练, 无法正常调试就直接使用水杉 vscode.

创建编译运行的json文件.

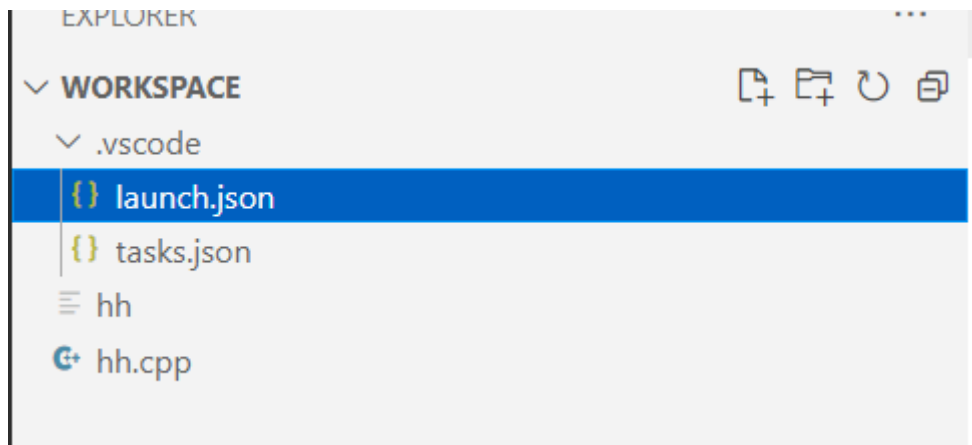
依次点击红框中的按键.



均选择第一项.



之后应该会在当前目录创建.vscode文件夹,其中包含launch.json和 tasks.json



修改配置文件来调试自己的代码.

json 文件

```
{
  "tasks": [
    {
      "type": "cppbuild",
      "label": "C/C++: g++ build active file",
      "command": "/usr/bin/g++",
      "args": [
        "-g",
        "${workspaceFolder}/hh.cpp", // 这个地方填要编译的文件。其中
        `${workspaceFolder}`为vscode打开的文件夹。
        "-o",
        "${workspaceFolder}/hh" // 这个地方填编译之后的可执行文件的地址和名字。
      ],
      "options": {
        "cwd": "${fileDirname}"
      },
      "problemMatcher": [
        "$gcc"
      ],
      "group": {
        "kind": "build",
        "isDefault": true
      },
      "detail": "Task generated by Debugger."
    }
  ],
  "version": "2.0.0"
}
```

launch.json 文件.

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?
  linkid=830387
  "version": "0.2.0",
  "configurations": [
```

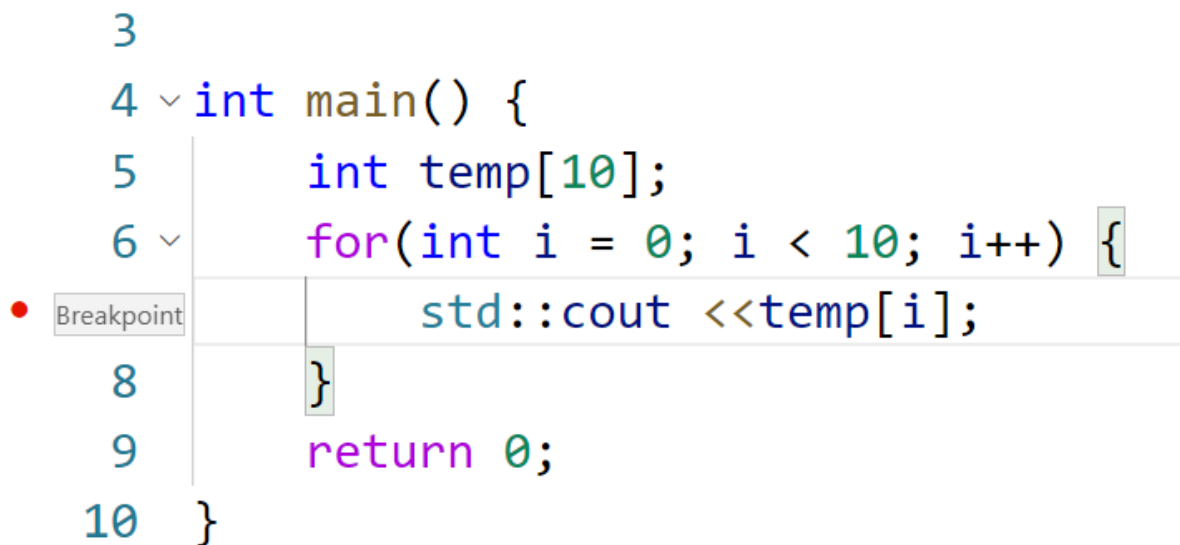
```

{
    "name": "g++ - Build and debug active file",
    "type": "cppdbg",
    "request": "launch",
    "program": "${workspaceFolder}/hh", // 这里填运行的二进制程序地址。
    "args": [],
    "stopAtEntry": false,
    "cwd": "${workspaceFolder}/", // 这里填运行二进制程序时候所在目录。一般不要改
    "environment": [],
    "externalConsole": false,
    "MIMode": "gdb",
    "setupCommands": [
        {
            "description": "Enable pretty-printing for gdb",
            "text": "-enable-pretty-printing",
            "ignoreFailures": true
        }
    ],
    "preLaunchTask": "C/C++: g++ build active file",
    "miDebuggerPath": "/usr/bin/gdb"
}
]
}

```

调试程序

可以在希望程序停止运行的地方打断点。



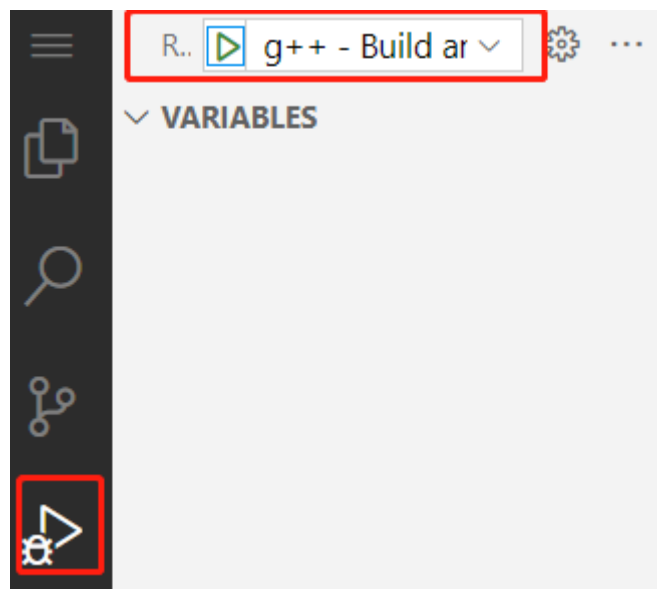
```

3
4  int main() {
5      int temp[10];
6      for(int i = 0; i < 10; i++) {
7          std::cout << temp[i];
8      }
9      return 0;
10 }

```

A red dot labeled "Breakpoint" is positioned to the left of line 7 in the code editor.

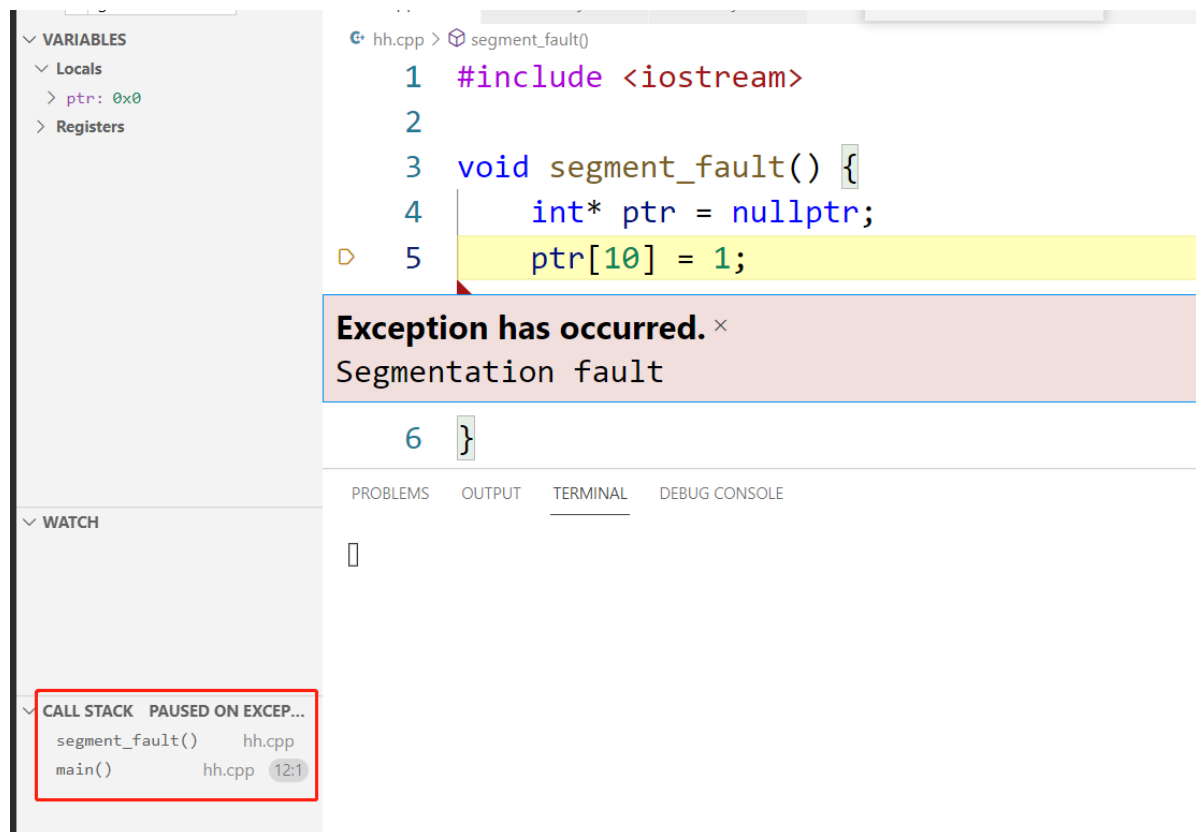
依次点击红框就可以运行程序了。



segmentation fault

比较常见的错误就是地址越界或者访问了非法地址.

可以参考左下角的调用栈定位出错位置.



变量使用时未初始化

可以看到变量未初始化时具有不确定性, 通常导致程序出现意外的结果.

hh.cpp > main()

3

4 int main() {

5 int temp[10];

6 for(int i = 0; i < 10; i++) {

7 std::cout <<temp[i];

8 }

9 return 0;

10 }

VARIABLES

Locals

i: 0

temp: [10]

[0]: 686803776

[1]: 32713

[2]: 0

[3]: 0

[4]: 2021370112

[5]: 22031

[6]: 2021369648

[7]: 22031

[8]: -1056124272

[9]: 32765

Registers

WATCH

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE