Homework3

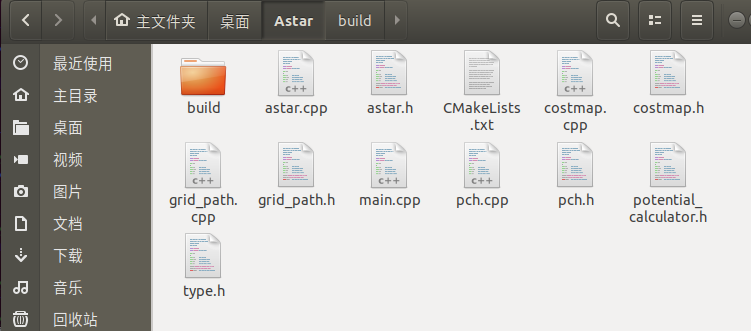
1. 参照所给A\*演示例程，修改代码，并在linux下编译，运行。

**方法一：**

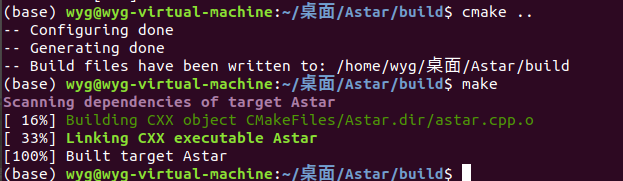
1.编写CMakeLists文件



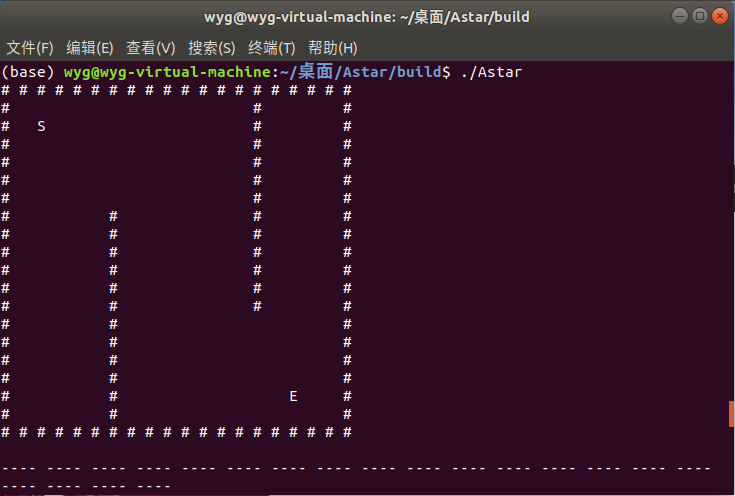
文件夹内容

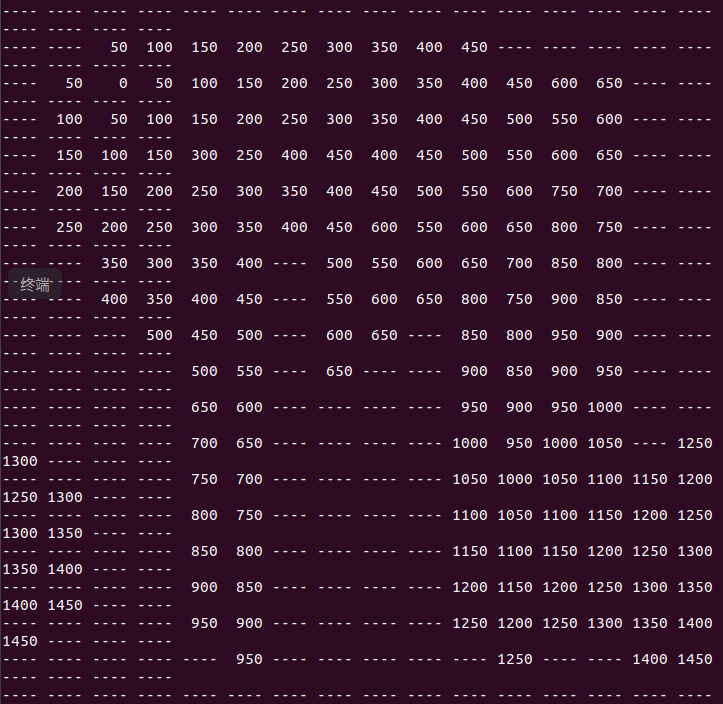


2.编译



3.运行

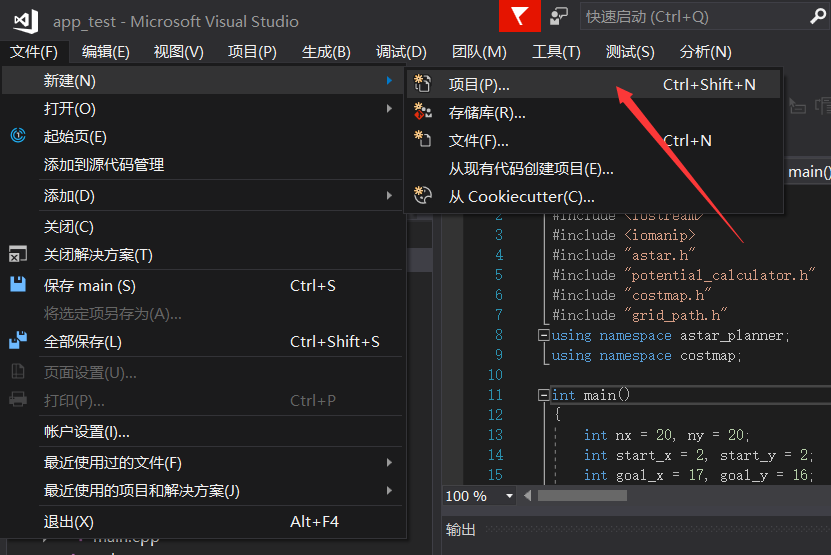




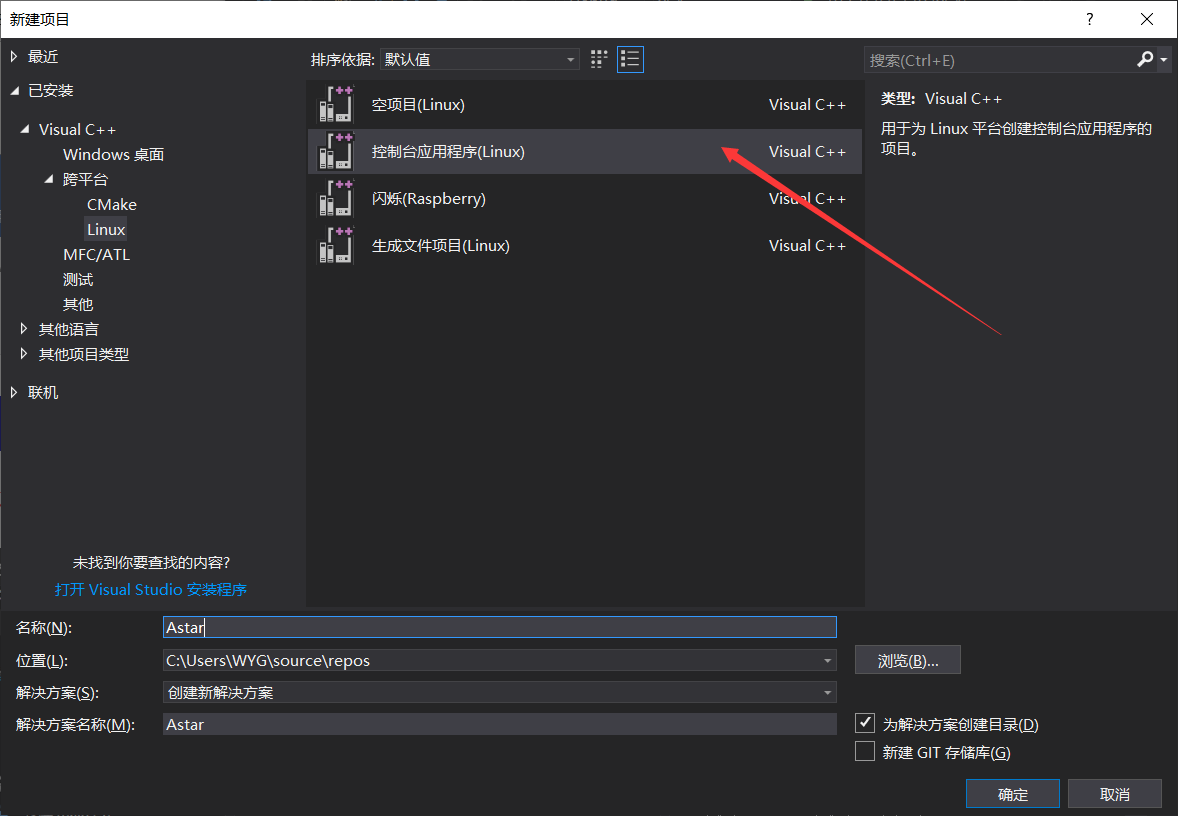


**方法二：**

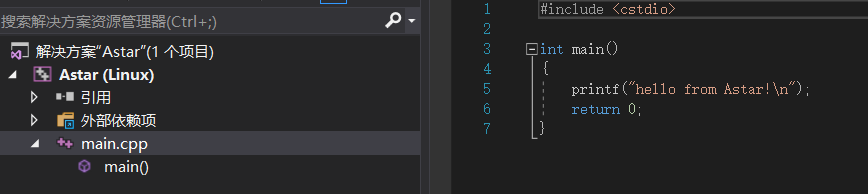
1.新建解决方案（跨平台Linux）：



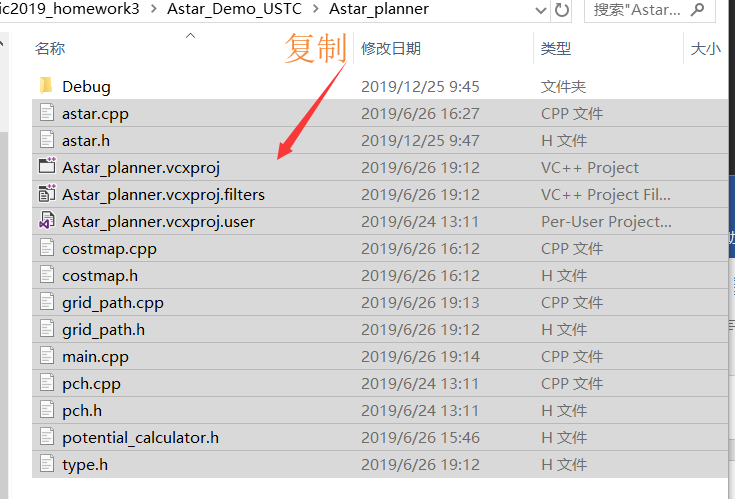
注意选择跨平台Linux，因为源程序时VS下编译的，现在我们要去Linux编译所以要跨平台，在win下的可执行程序是.exe而在Linux下是.out或者没有。

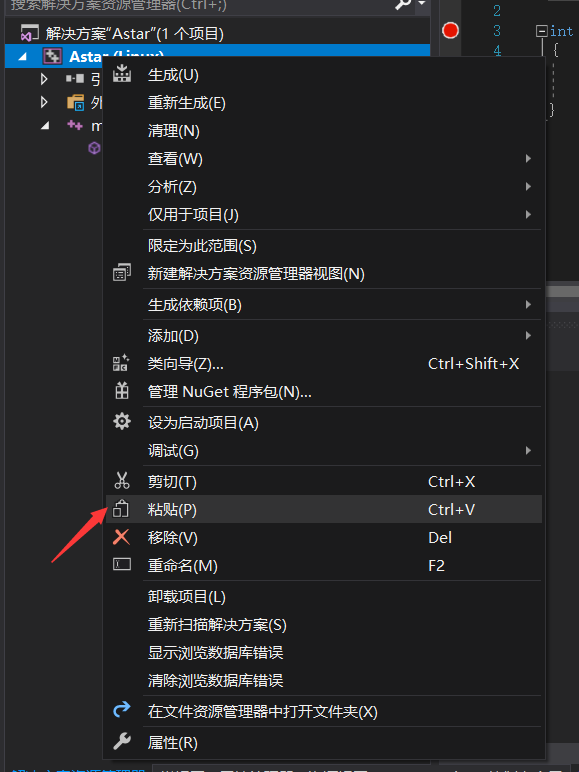


新建完项目会自带一个main函数

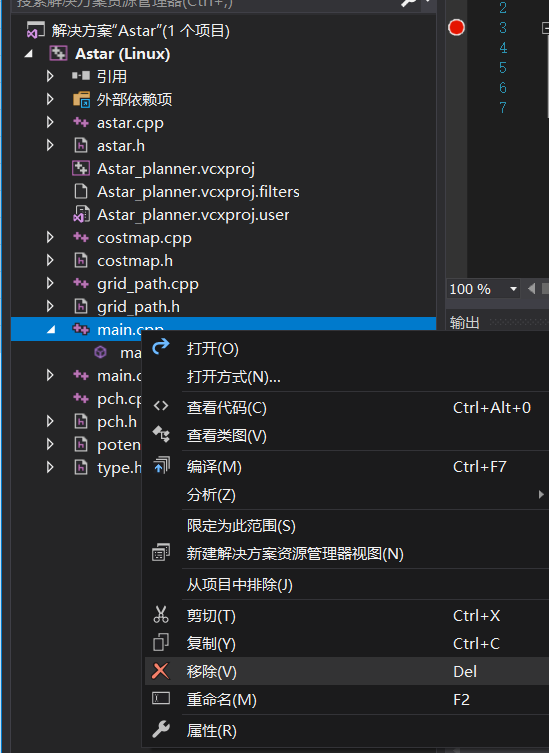


我们不用这个函数，现在去复制源码中的源文件和头文件，粘贴到项目下面

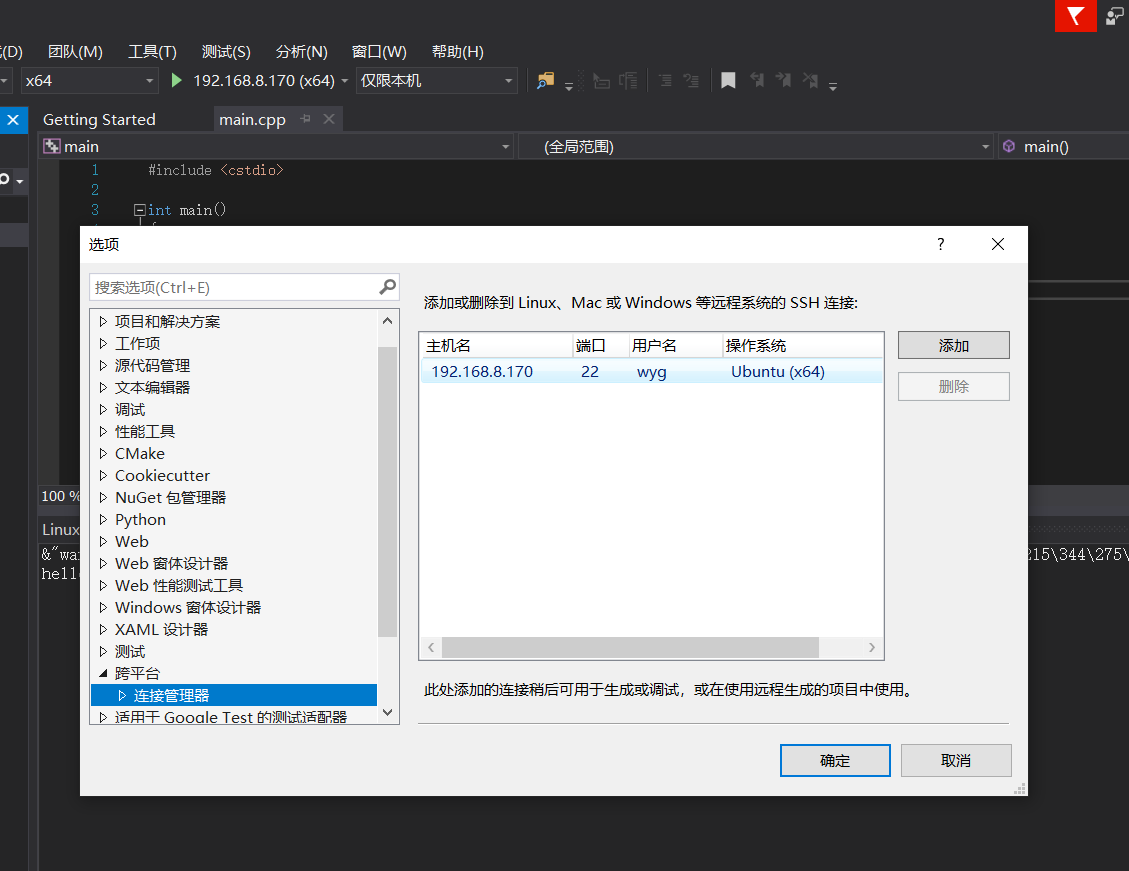




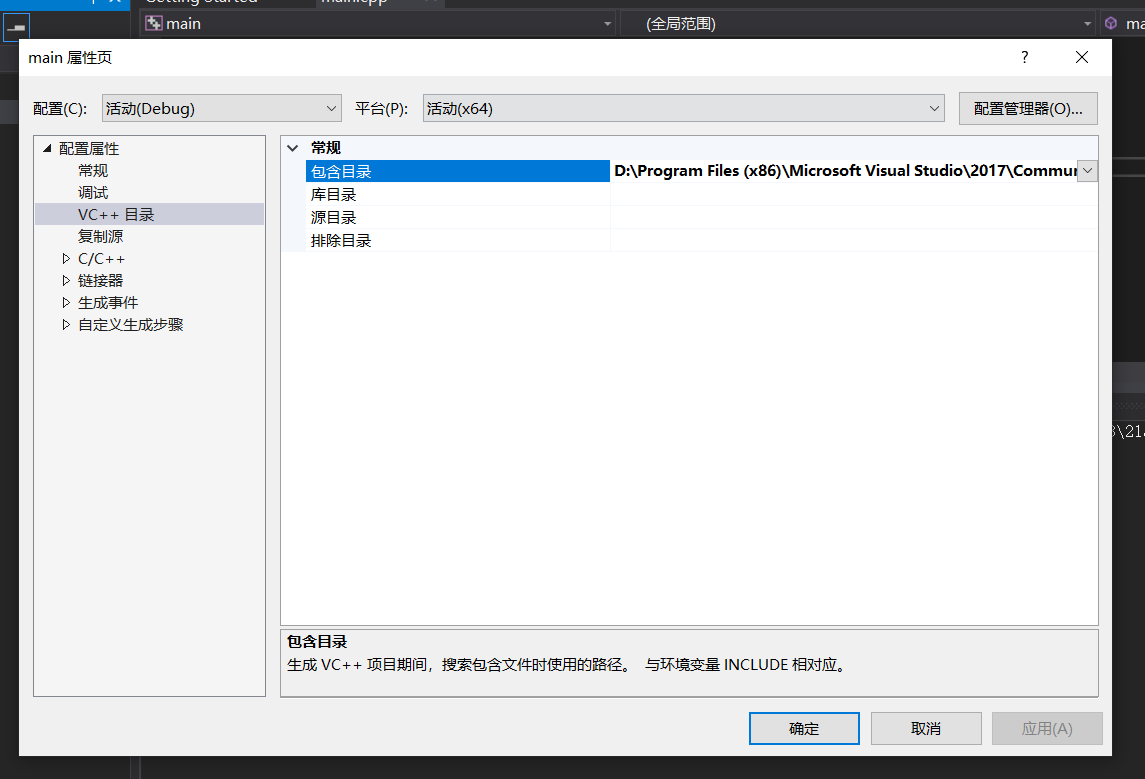
再删除原有的main函数



点击工具，选项设置连接管理器，连接到了我的ubuntu虚拟机的地址，并保存了用户名和密码



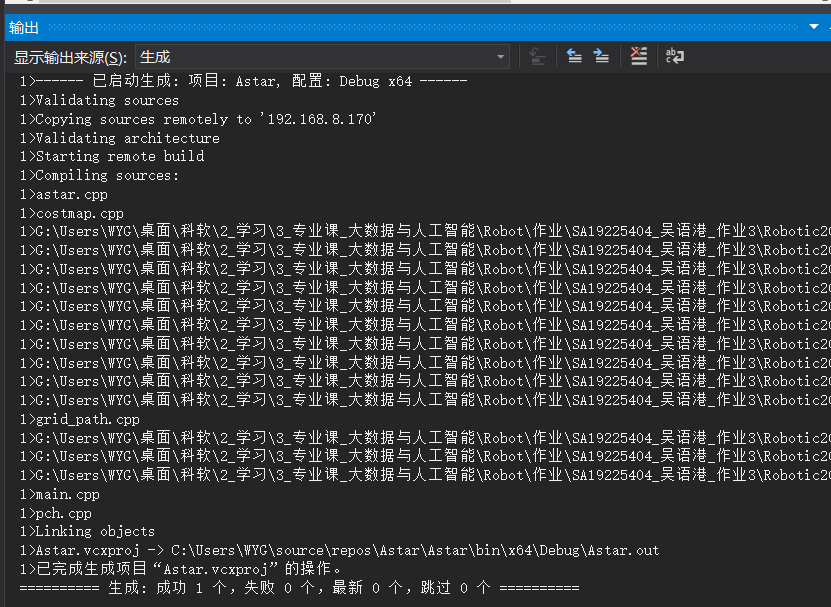
再添加一下头文件的路径D:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\IDE\VC\Linux\include\usr;$(IncludePath)



生成解决方案



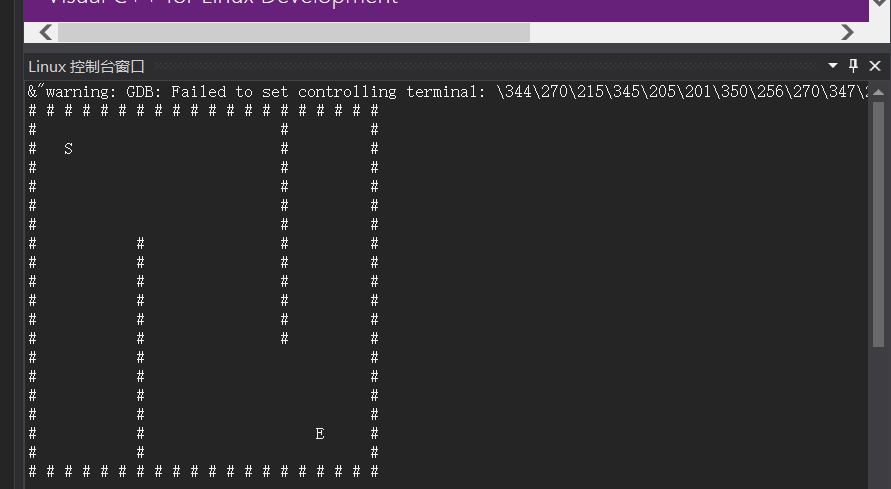
编译完成

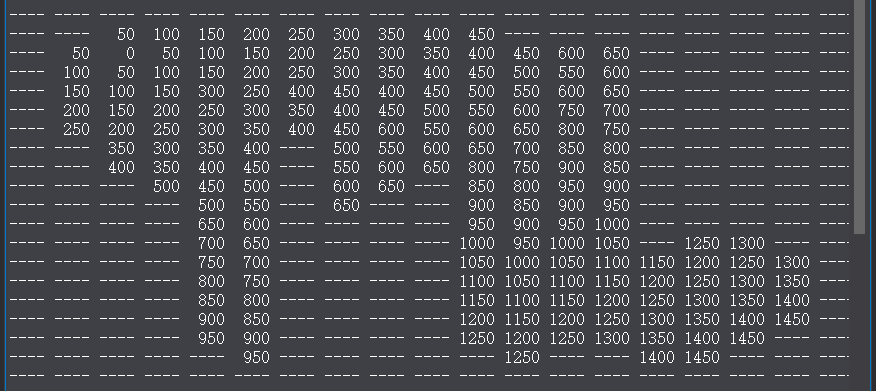


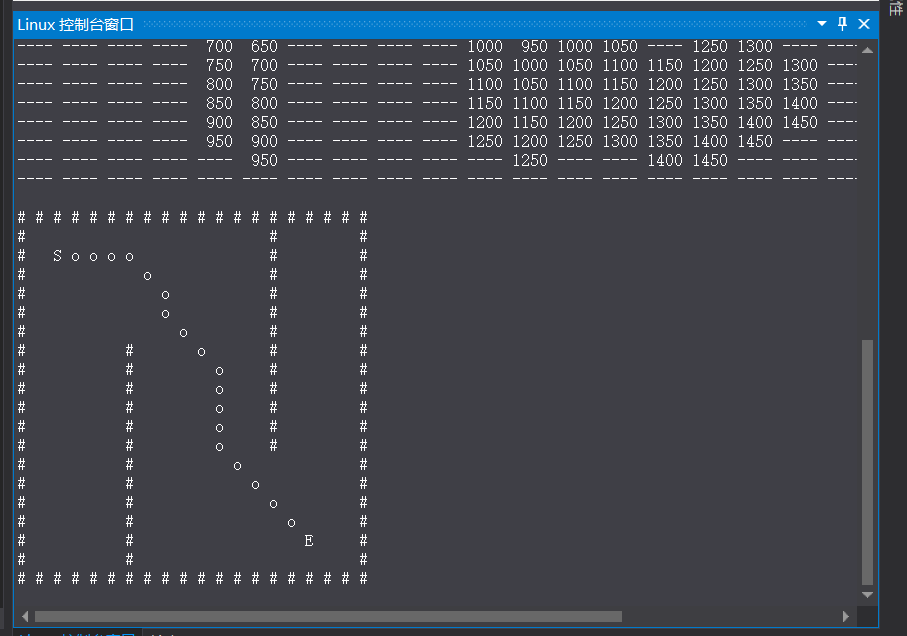
开始调试，点击调试按钮



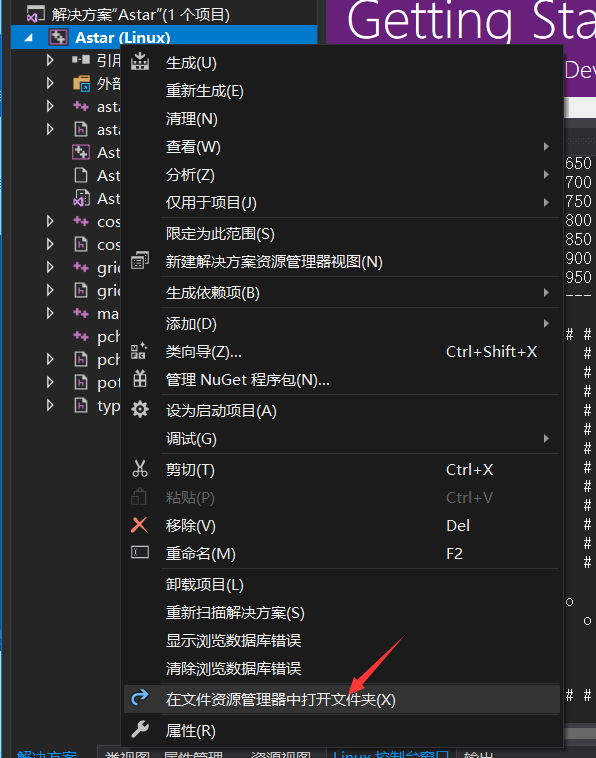
地图



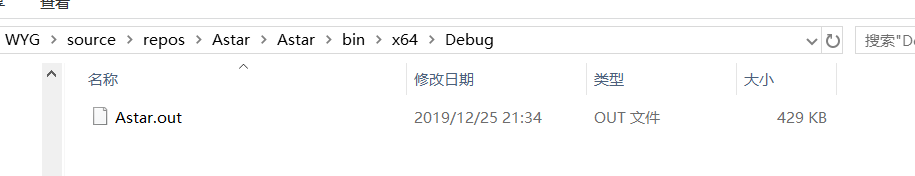




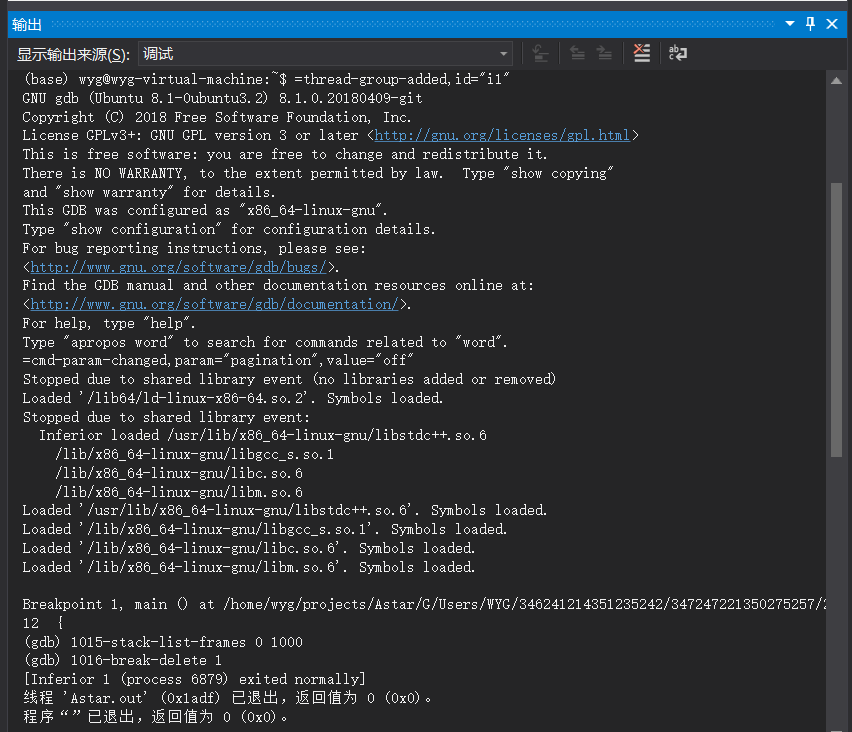
查看项目目录



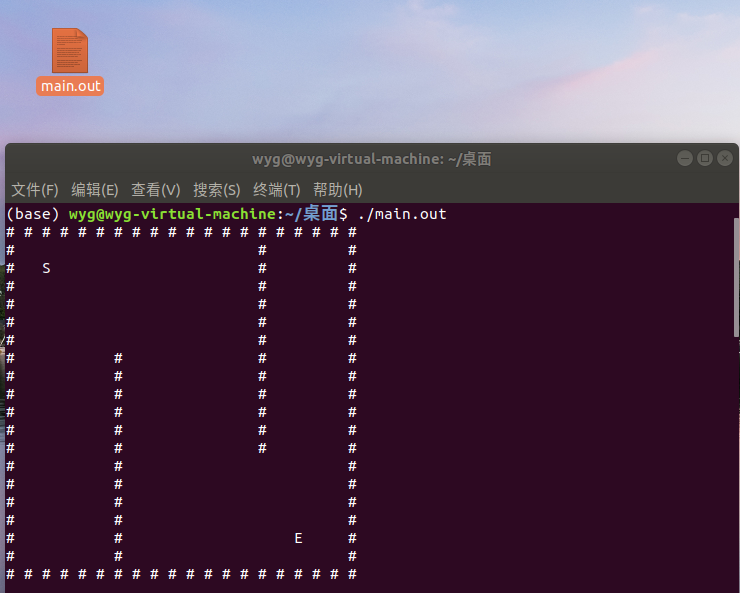
生成了.out文件

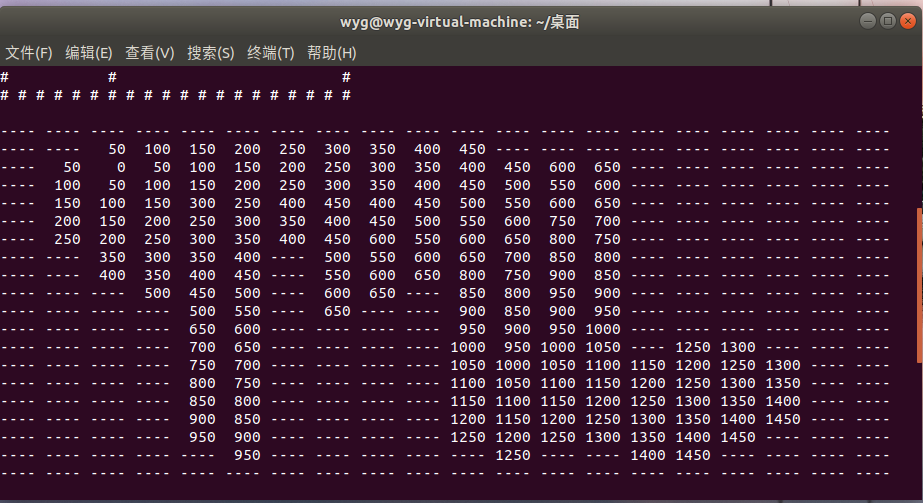


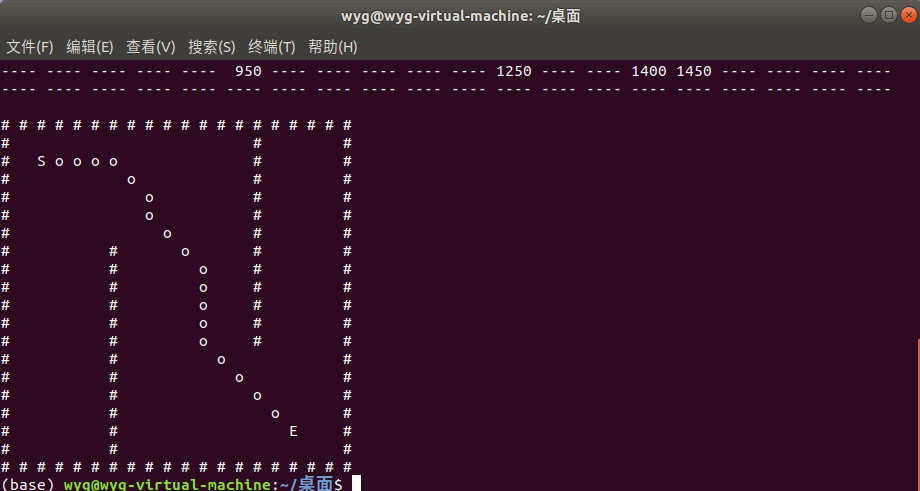
调试的输出信息



将.out文件复制到ubuntu里面运行







说明Linux下编译运行的问题搞定了

二. 结合课堂所介绍的A\*算法，栅格地图中两点距离度量，修改代价函数，运行并分析演示效果。

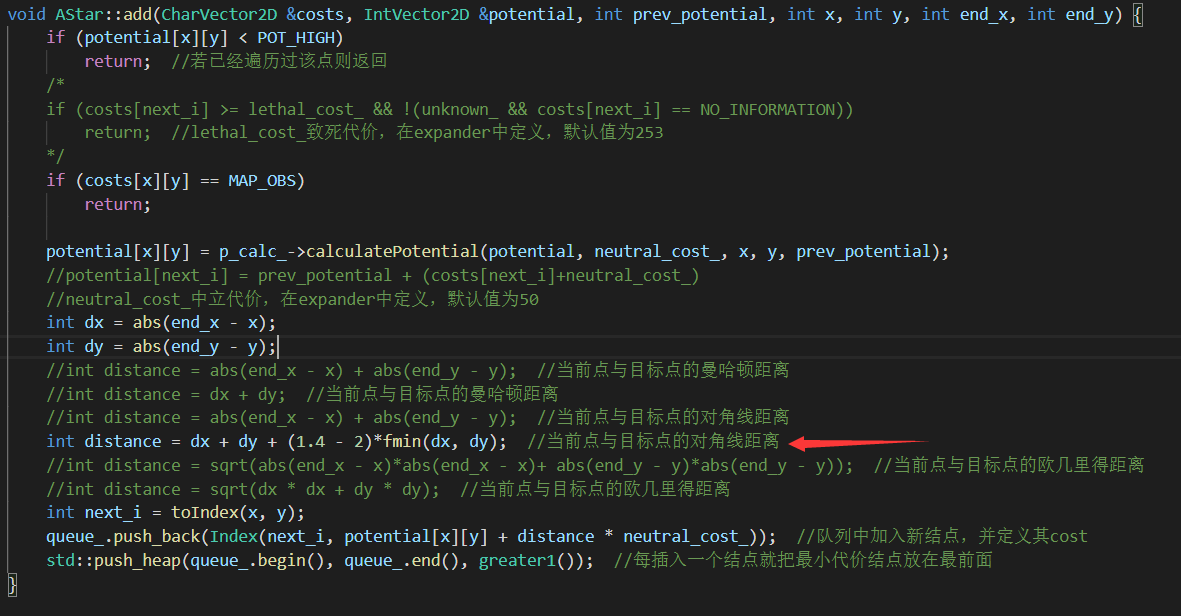
一、修改为对角线距离（修改Astar.cpp中的启发路径语句）

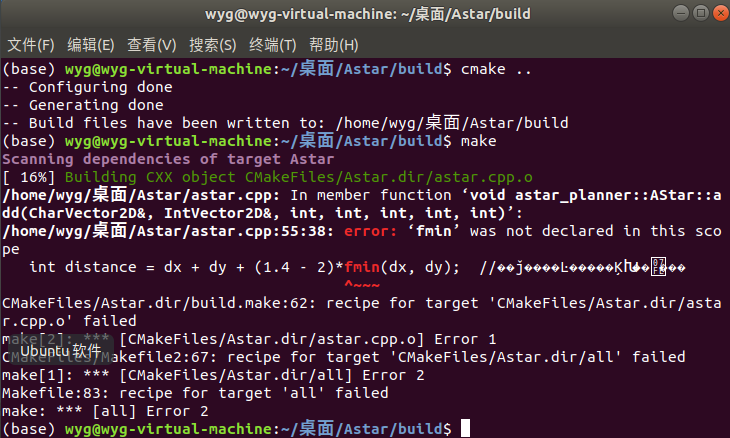
1.运行（修改启发函数采用的距离为对角线距离）

int dx = abs(end\_x - x);

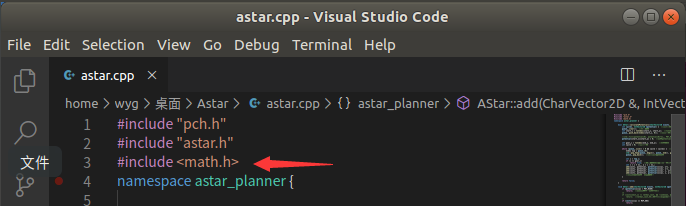
int dy = abs(end\_y - y);

int distance = dx + dy + (1.4 - 2)\*fmin(dx, dy);  //当前点与目标点的对角线距离

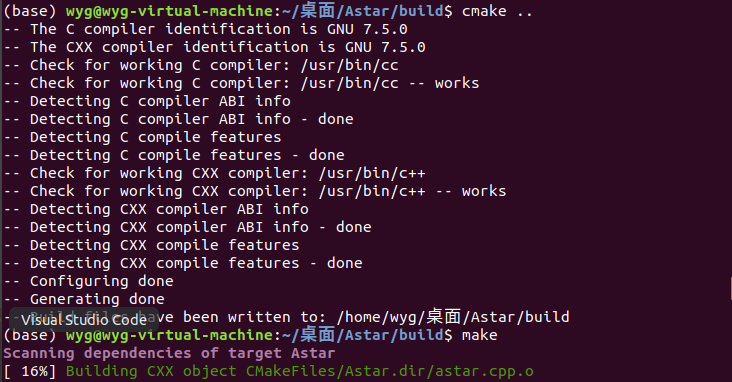


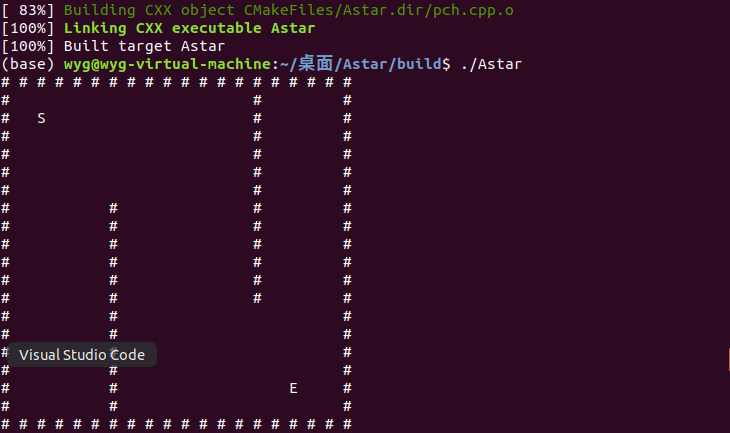


编译出错，看来是没有把math库包含进去，去添加

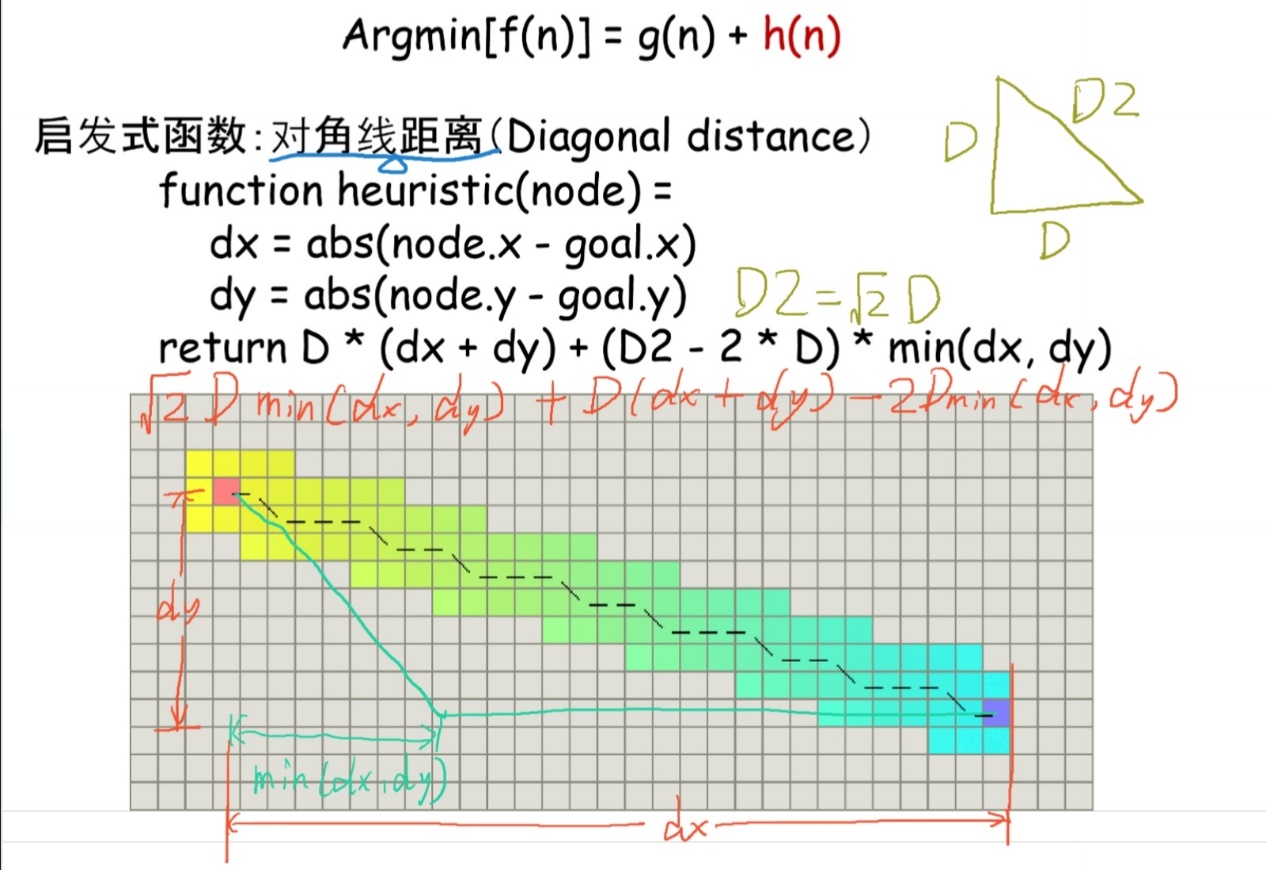


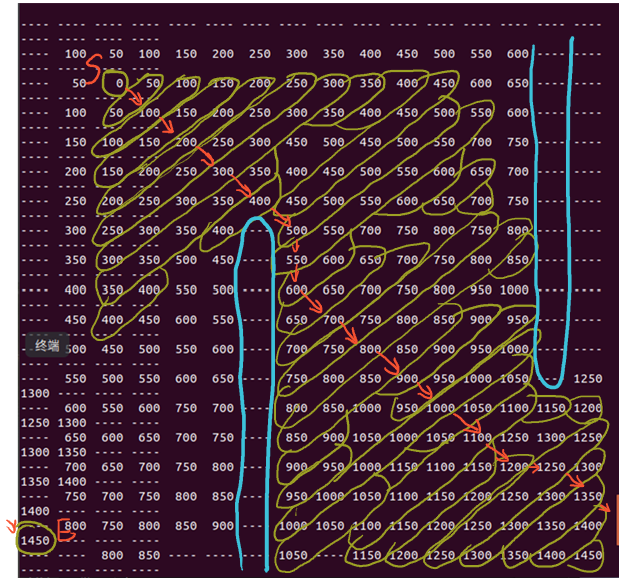
不知道为什么，window不用加这个也能运行





2.分析





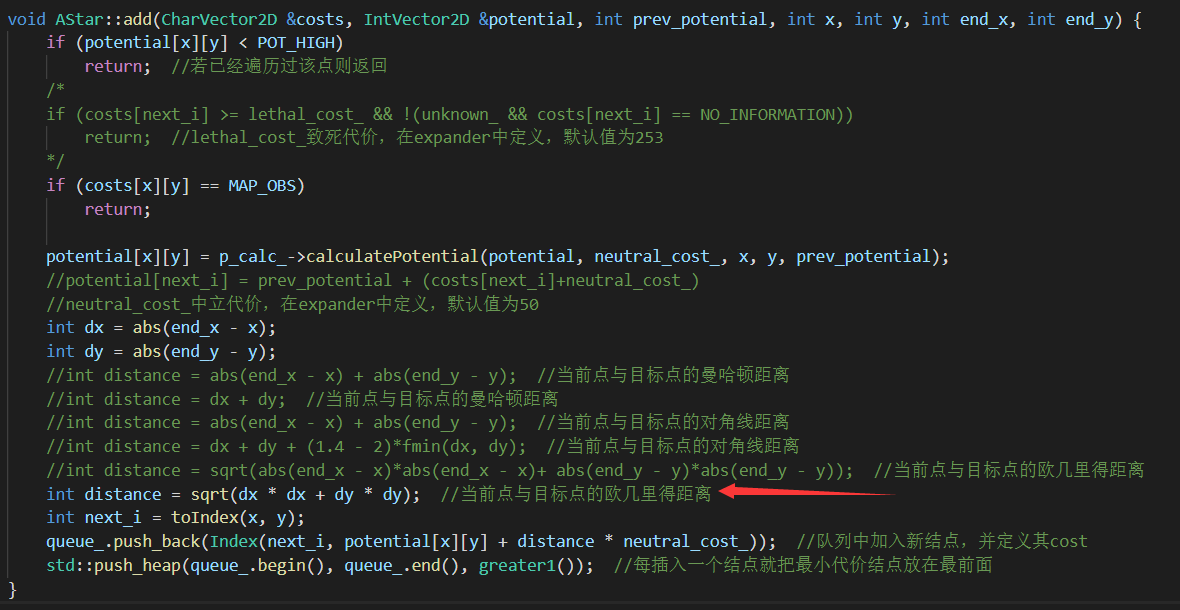
途中的路径为在对角线距离启发函数下搜索出的最佳路径，绕过了中间的一些奇怪（高地，可能是因为搜素算法导致的问题）。

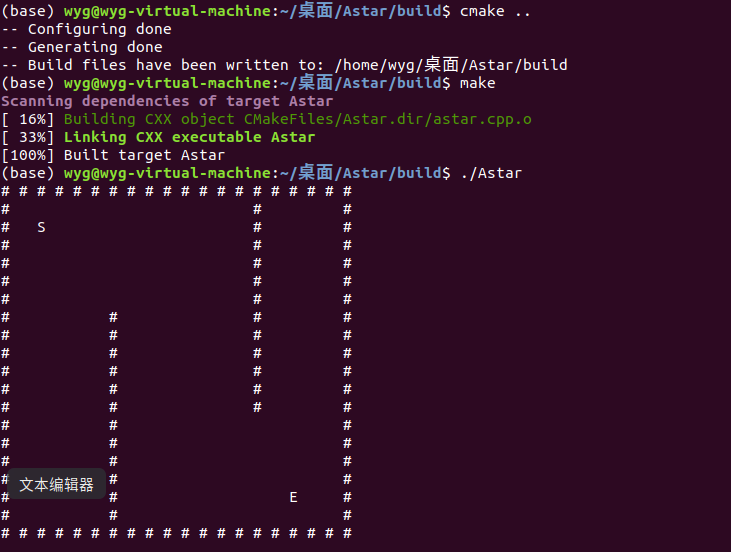


二、修改为欧几里得距离

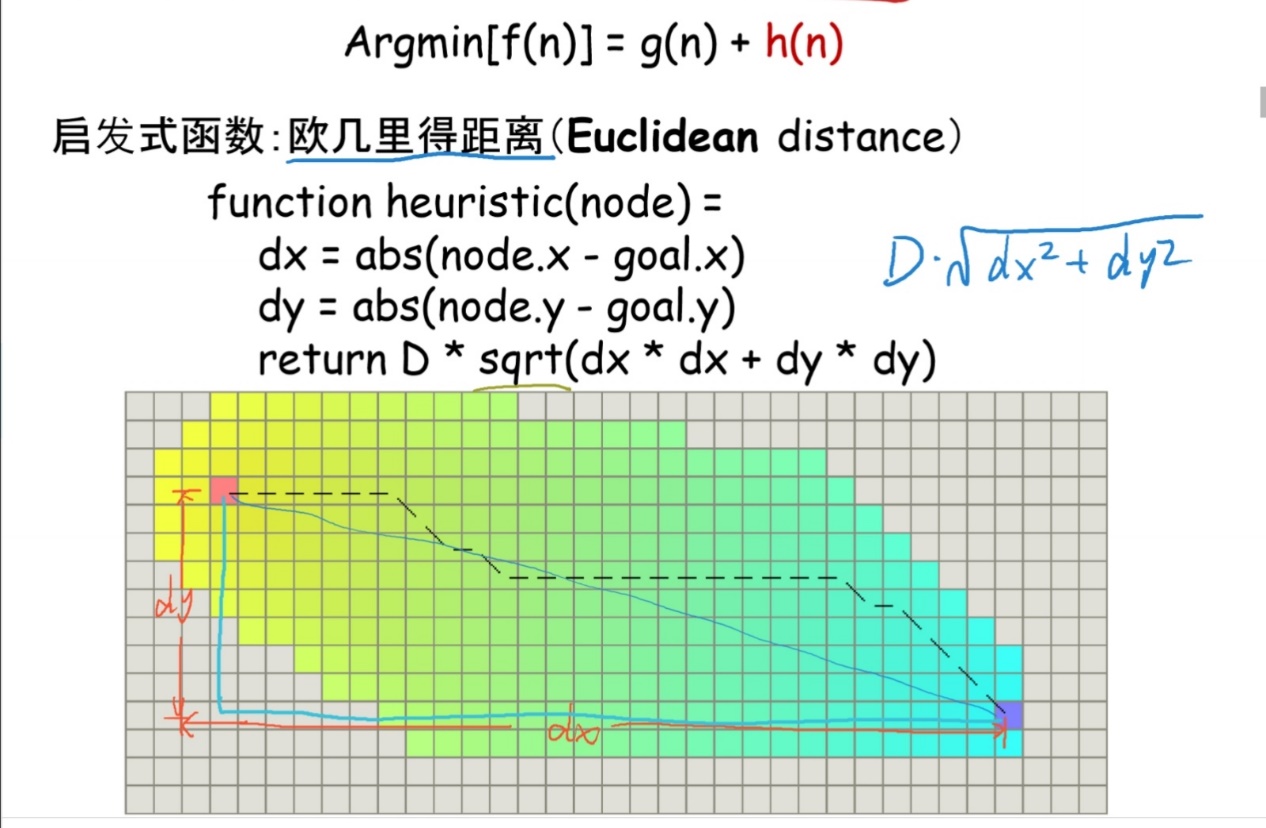
1.运行（修改启发函数采用的距离为欧式距离）

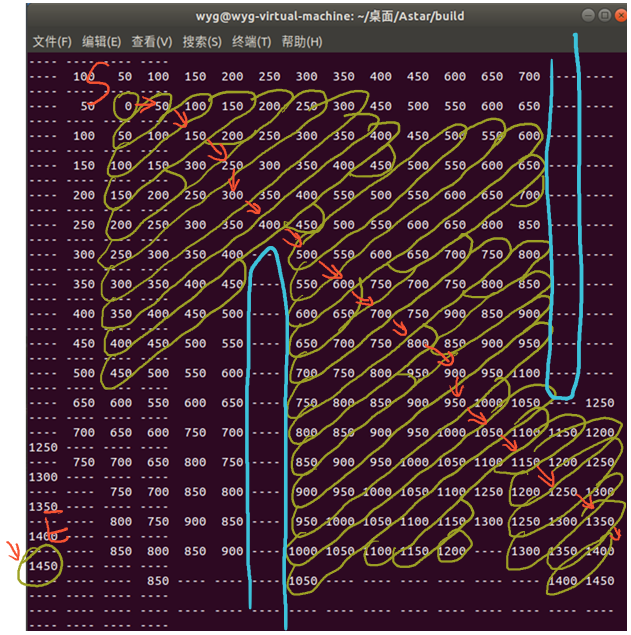
int distance = sqrt(dx \* dx + dy \* dy);  //当前点与目标点的欧几里得距离





2.分析





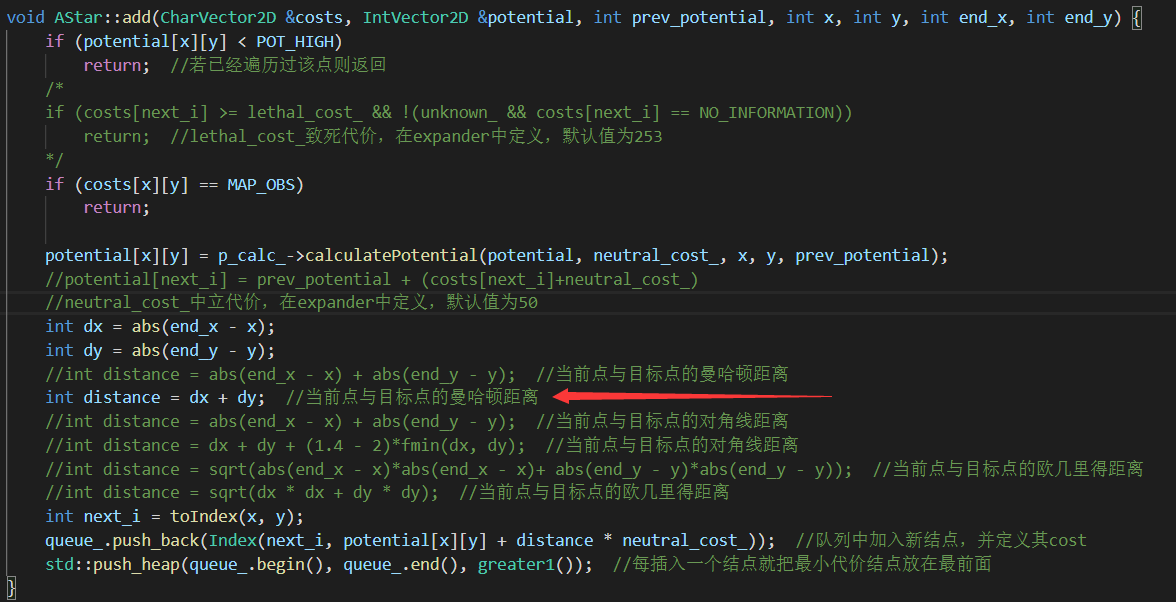
途中的路径为在欧几里得距离启发函数下搜索出的最佳路径，绕过了中间的一些奇怪（高地，可能是因为搜素算法导致的问题）

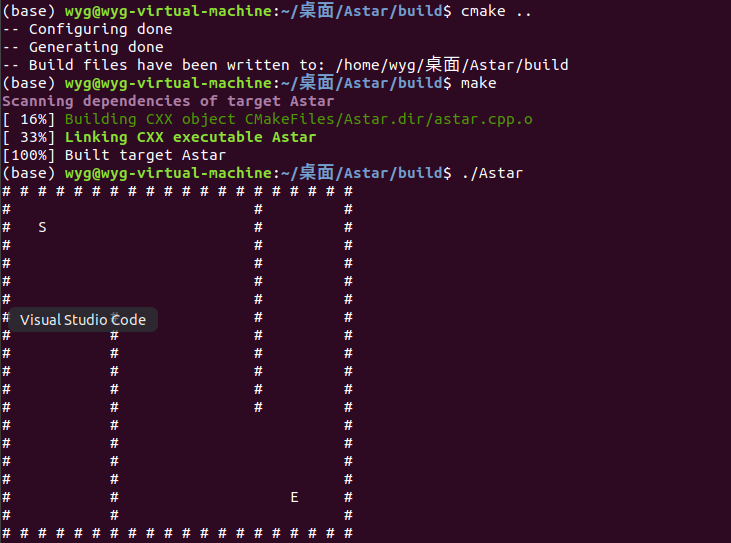


三、原本为曼哈顿距离

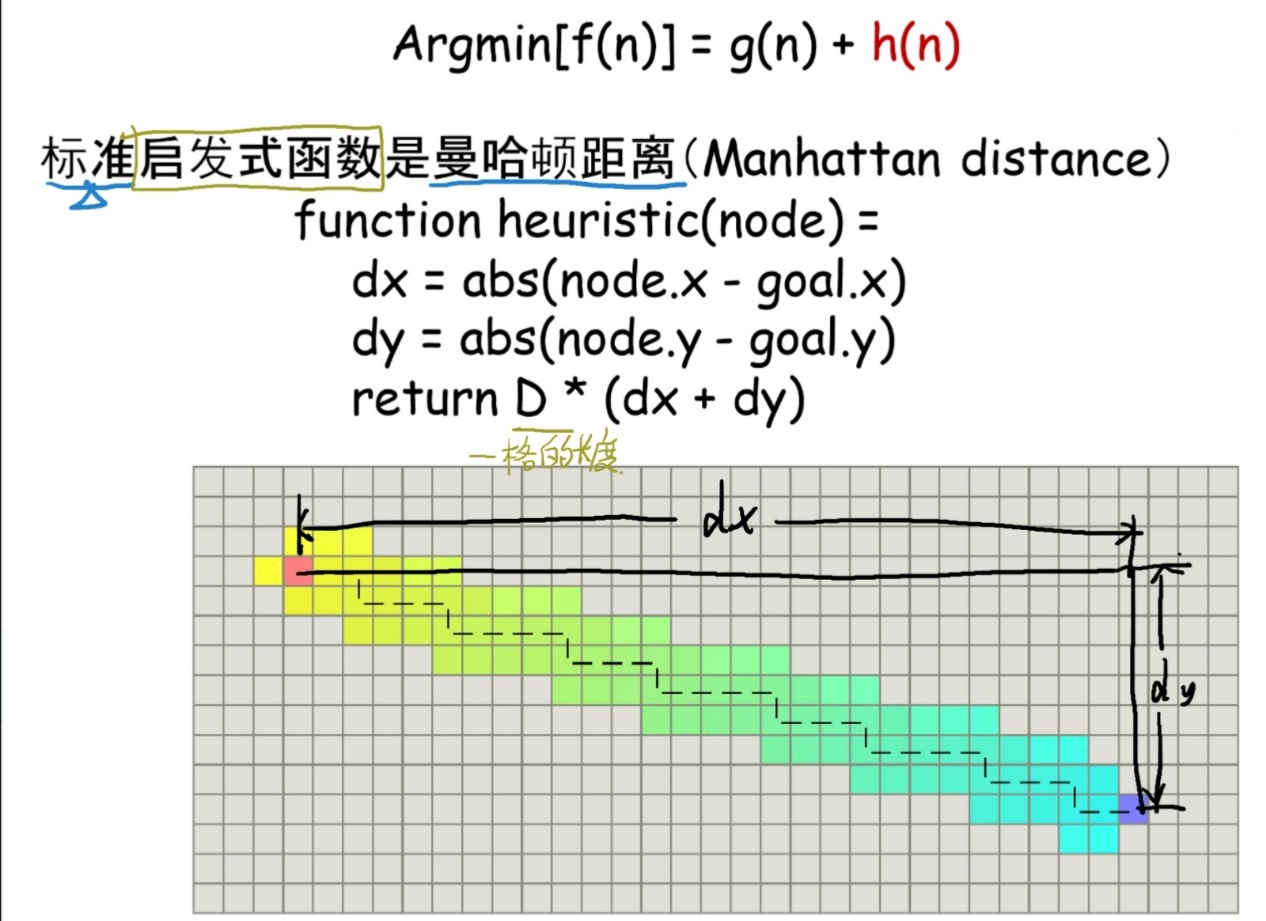
1.运行（修改启发函数采用的距离为曼哈顿距离）

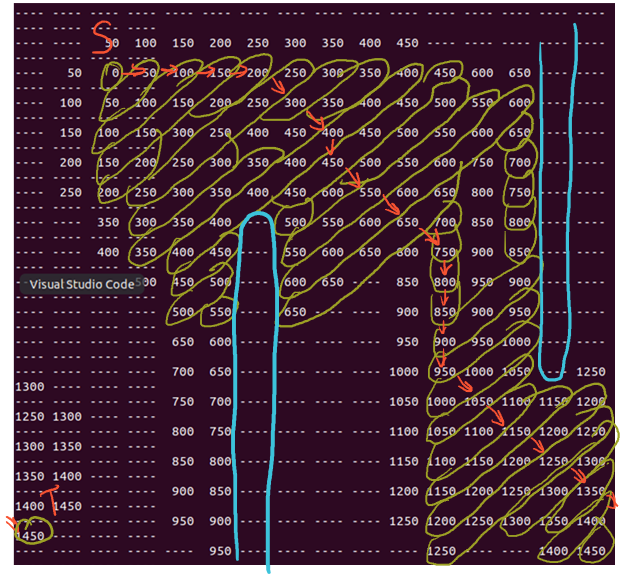
int distance = dx + dy;  //当前点与目标点的曼哈顿距离





2.分析





途中的路径为在曼哈顿距离启发函数下搜索出的最佳路径，绕过了中间的一些奇怪（高地，可能是因为搜素算法导致的问题）

