**C++开发环境Codelite使用说明**

Codelite是一款跨平台的、轻量的C/C++开发环境，安装使用方便。

目前，其最新版本是V16，但是由于其不带编译器，所以建议安装V6.1.1，即[codelite-6.1.1-mingw4.8.1.exe](https://sourceforge.net/projects/codelite/files/Releases/codelite-6.1/codelite-6.1.1-mingw4.8.1.exe/download)，该版本自带Mingw编译器（Windows平台上的GNU C++编译器）。

**1.下载**

用户可以从如下网址下载Codelite。

网址1：

<https://fsn7uk3il5.feishu.cn/file/boxcn6v8XJE884YRLsnBor5ZyCd>

网址一：

<https://sourceforge.net/projects/codelite/files/Releases/codelite-6.1/>

从网址1下载，如果下载失败，只需在浏览器上回退，然后尝试重新点击下载即可。

**2.安装**

双击安装文件“[codelite-6.1.1-mingw4.8.1.exe](https://sourceforge.net/projects/codelite/files/Releases/codelite-6.1/codelite-6.1.1-mingw4.8.1.exe/download)”，开始安装。

同意协议，下一步。

同意Codelite缺省目录，下一步。

采用全部安装模式，下一步，如图1所示。

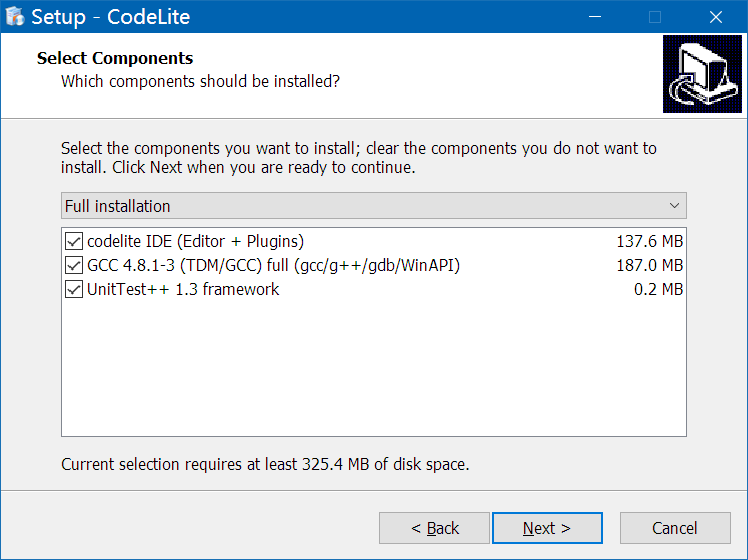


图1. 全部安装

同意UnitTest缺省目录，下一步。

同意Mingw缺省目录，下一步。

同意Codelite程序组名称，下一步。

点击创建桌面图标“create a desktop icon”，下一步，如图2所示。

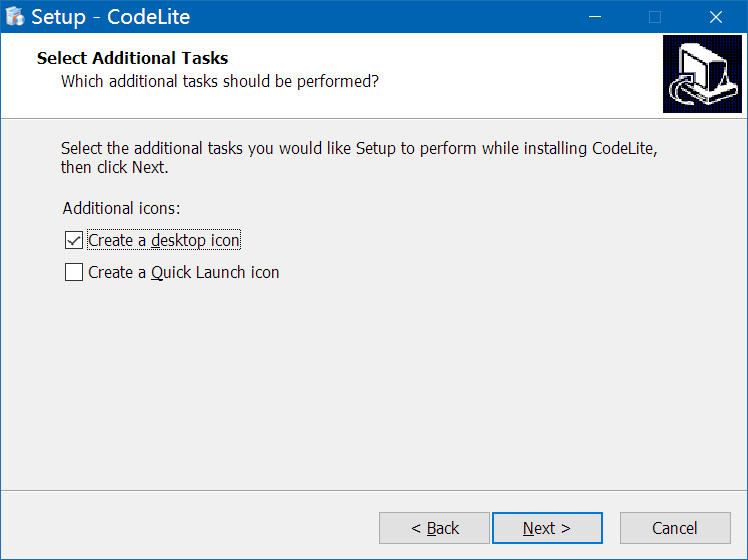


图2. 创建桌面图标

开始安装，等待安装完毕即可。

**3.新建workspace, project**

开始🡪程序🡪…Codelite，点击快捷键，运行Codelite。

第一次运行需要配置编译器，选择mingw即可，如图3所示。

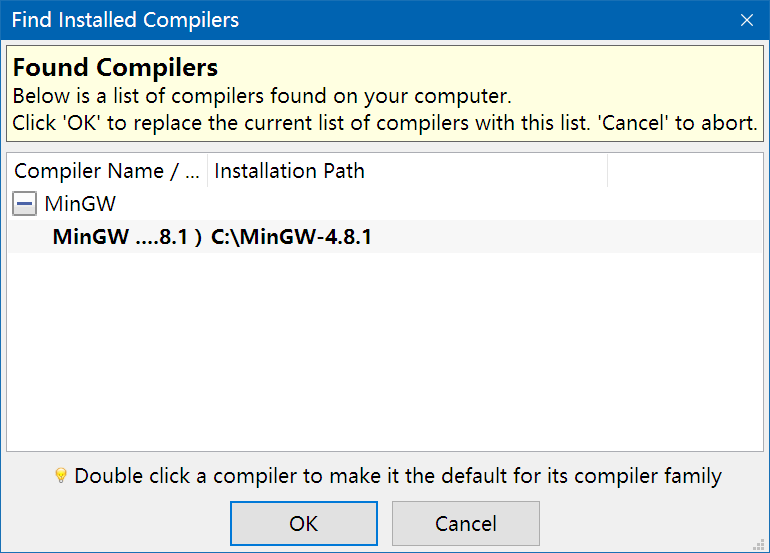


图3.选择编译器

提示reset tool bar，选择yes。

Codelite的各个视图(View)是可以停靠在主窗口的四个边的，我们一般采用如下的Layout，即布局。Workspace视图在左，Debugger视图在右，Output视图在下，中间用于打开和编辑代码。如图4所示。

用鼠标拖拽各个视图（Workspace视图、Output视图、Debugger视图）的标题栏即可移动和停靠。

如果Debugger视图没有打开，可以通过“View🡪Debugger Pane”菜单打开。

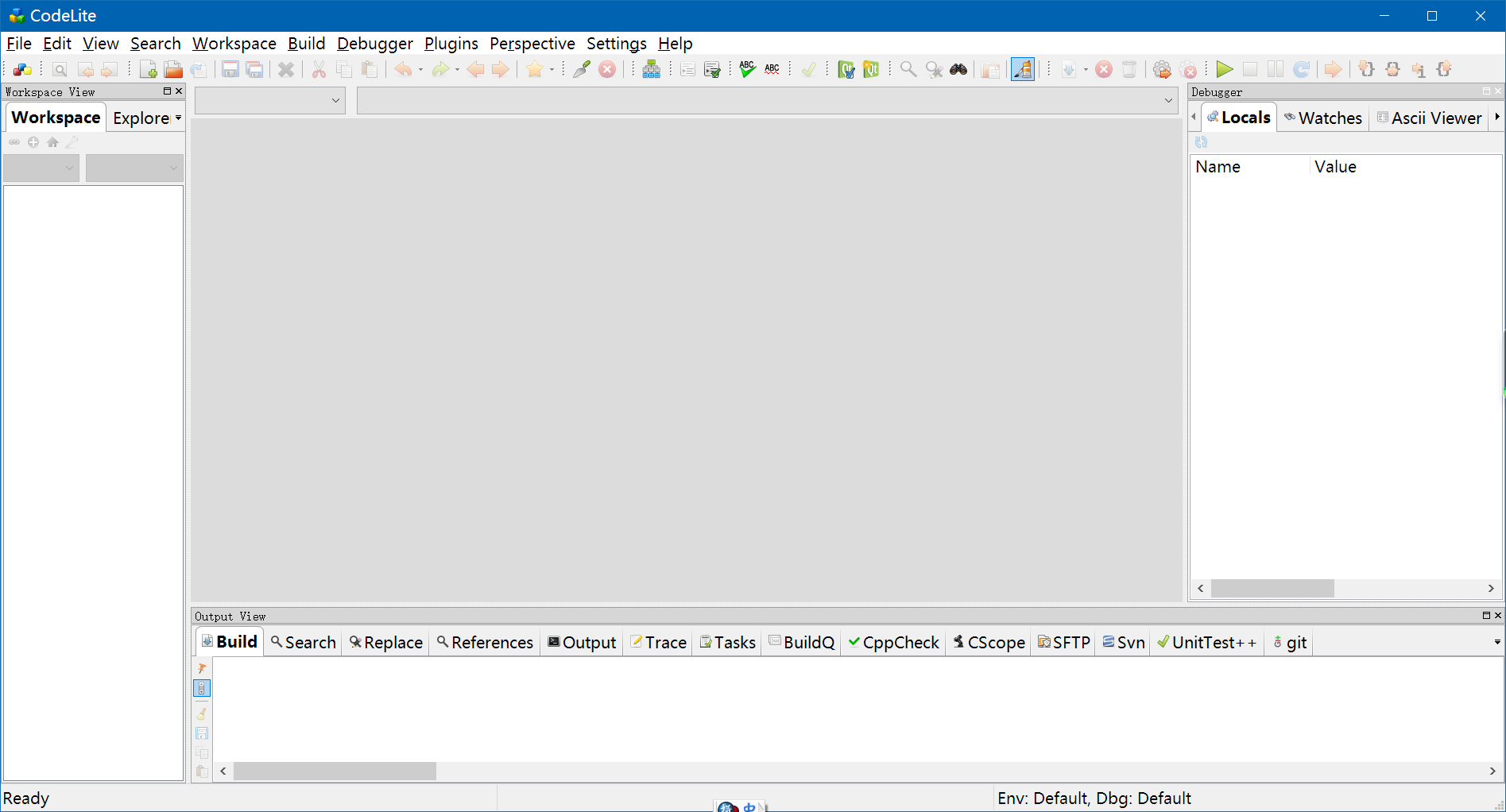


图4. 开发环境布局

在D:盘(没有D:盘在C:盘也可以)，通过资源管理器，新建D:\homework目录。

在Codelite里，通过“File🡪New🡪New workspace”菜单，新建一个名称为“workspace2019”的workspace（即工作空间），工作空间所在目录为“D:\homework”，如图5所示。

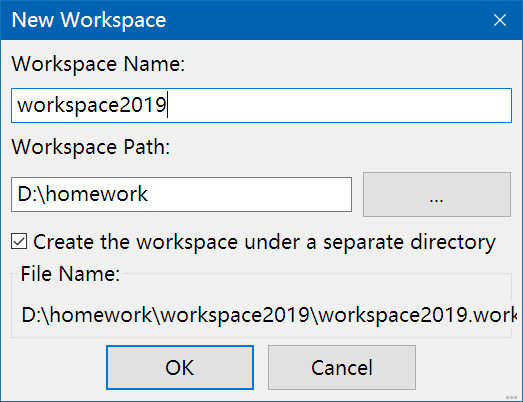


图5. 创建工作区

选择Workspace视图的“workspace2019”工作空间，如图6所示。

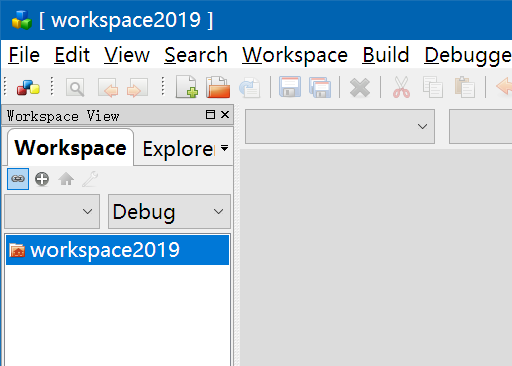


图6. 选择工作区

通过“File🡪New🡪New project”菜单，开始新建一个项目，项目类型为“console🡪simple executable(g++)”，如图7所示。

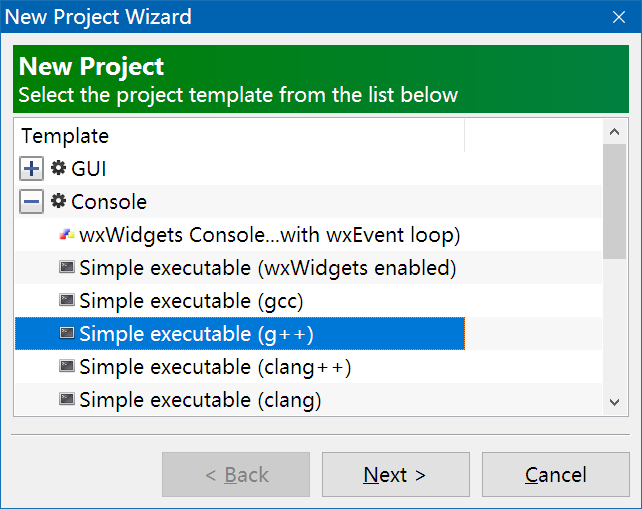


图7. 新建的项目类型

下一步，项目名称为“project01”，项目所在目录为“D:\homework\workspace2019”，如图8所示。

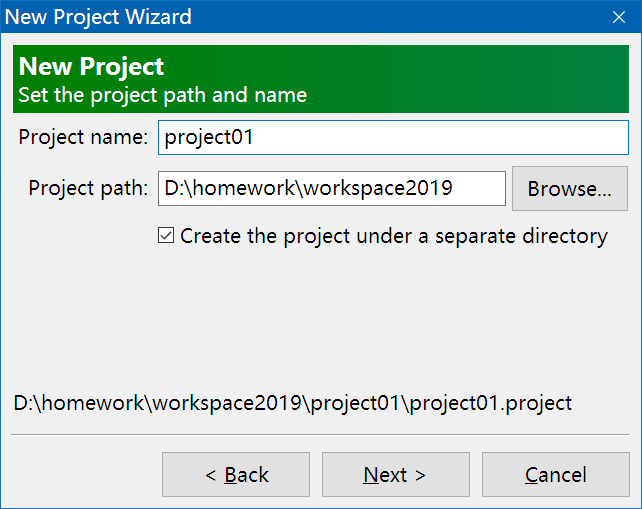
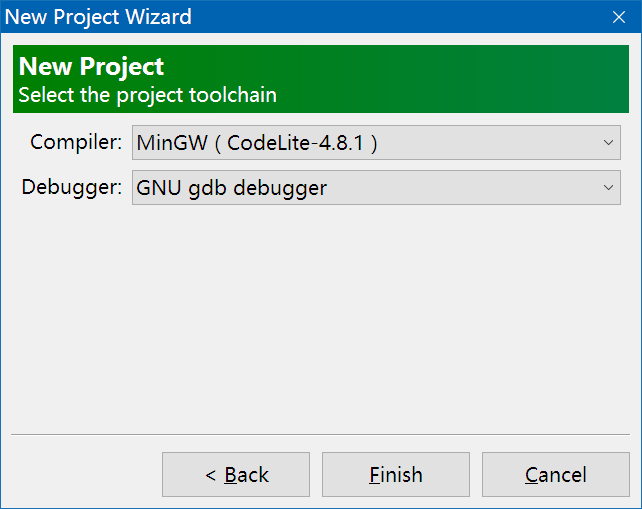


图8. 新建项目名称和存盘目录

选择缺省的编译器和调试器即可，如图9所示。



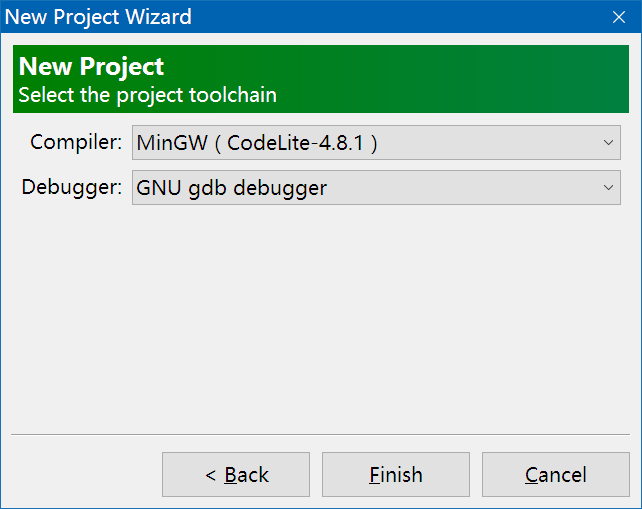


图9. 选择项目的编译器和调试器

至此，我们已经生成一个工作空间，该工作空间包含一个项目。

在Workspace视图，点击“+”号打开“project01”，继续点击“+”号打开“src”，双击“main.cpp”，打开源代码，这是Codelite为我们准备好的一个程序模板，可以直接运行。

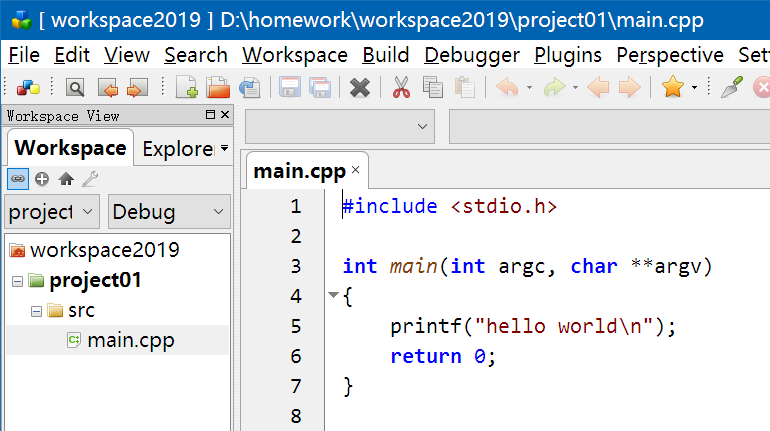


图10. 打开代码

我们添加一些代码，如图11所示。功能是定义两个整型变量，给其赋值，然后相加，结果赋给第三个整型变量，最后打印输出。

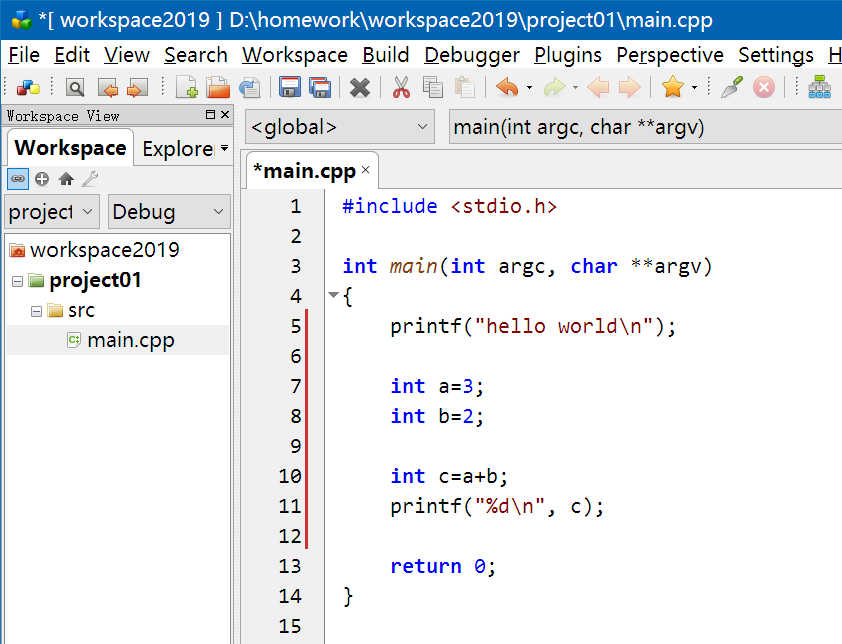


图11. 缺省代码

注意： 每个程序都是一个项目。工作空间和项目的关系是，一个工作空间包含多个项目。如果要新建一个程序，需要在当前工作空间建立新项目，通过菜单“File🡪New🡪New Project”完成。

**4.运行和调试代码**

选择“Build🡪Run”菜单，运行程序。

在出现的提示框，选择“Build and Execute”，程序即被编译和运行，输出结果如图12所示。

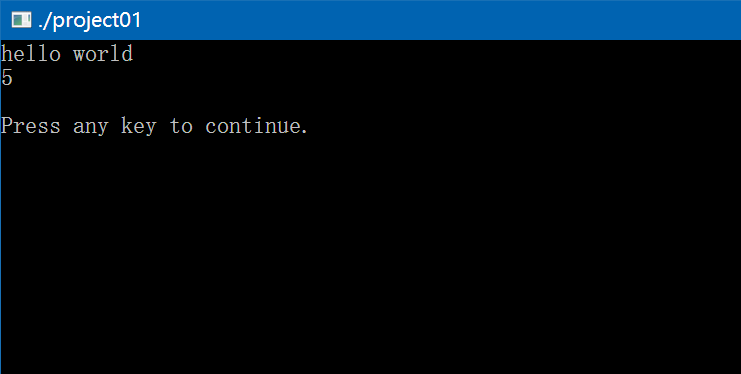


图12. 运行

在“int a=3;”代码的左侧，行号右边，按鼠标左键，加上一个断点（程序执行过程中，调试器临时暂停的代码行）。

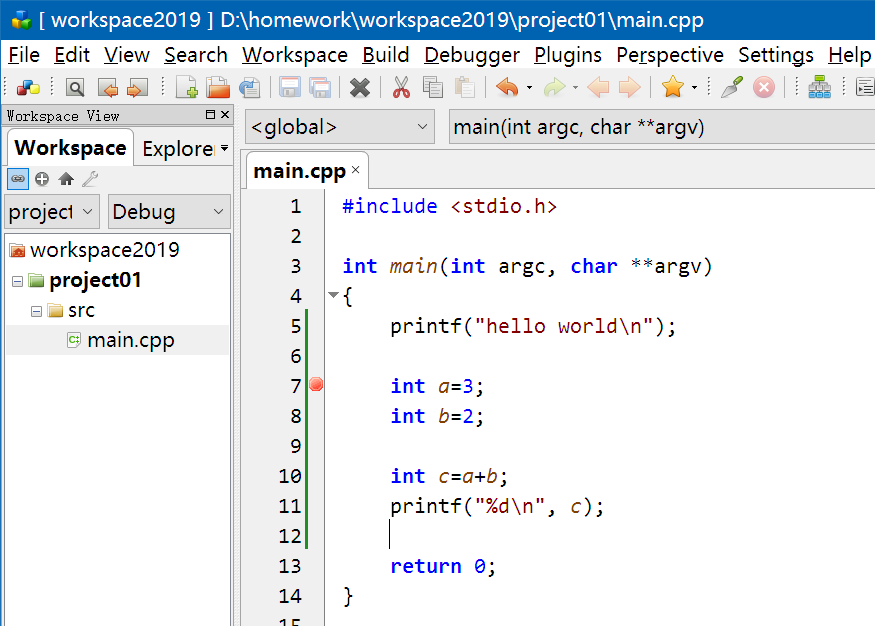


图13.设定断点

选择“Debugger🡪Start/Continue Debugger”菜单，开始调试代码。

在提示框，选择“Build and Debug”。

代码执行到断点处，出现绿色的向右箭头，如图14所示。

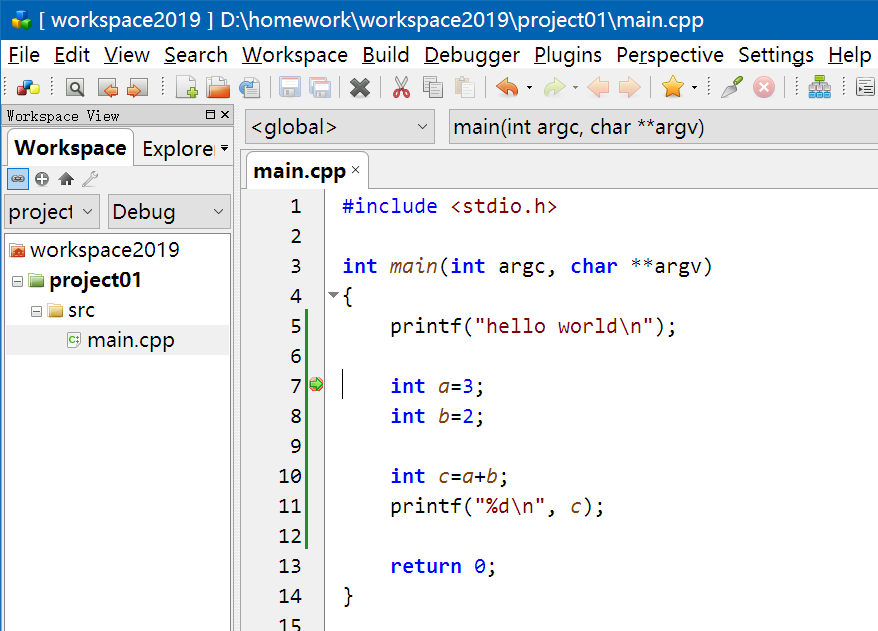


图14. 运行到断点

选择工具栏上的“Next”图标，执行本行代码，到下一行代码停住，如图15所示。

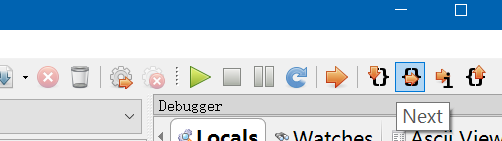


图15. Next按钮

这时候，Debugger面板上的局部变量的值发生改变，a已经被赋值了。

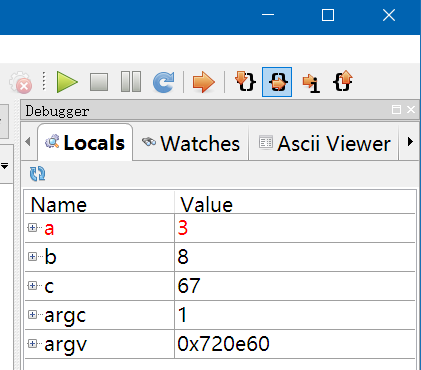


图16.变量的值

通过一步步执行代码，和我们预期的变量的值做比较，如果程序执行的结果和我们的预期，核对后有差异，就可以发现程序中有Bug的代码，然后排除这些Bug。

当我们一步步执行到“printf("%d\n", c);”，我们发现Debugger面板上，局部变量的值发生了预期的改变，如图17所示。

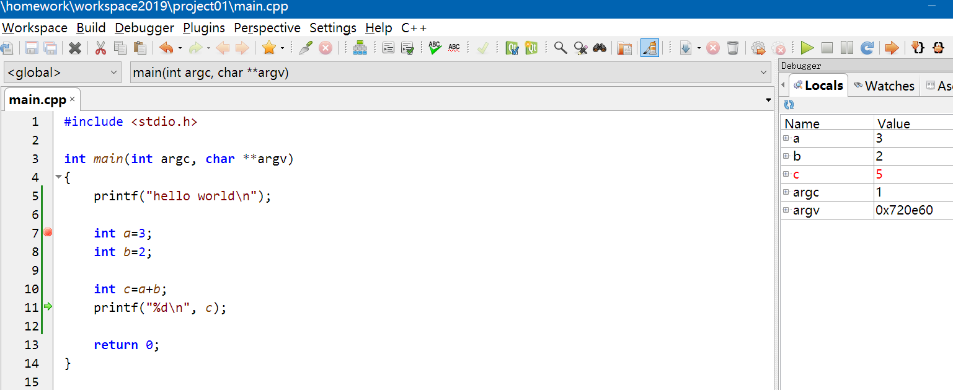


图17. 单步执行

如果我们想进入某个函数，跟踪调试其中的代码，当调试器停留在函数调用行的时候，我们应该选择“Step In”按钮。

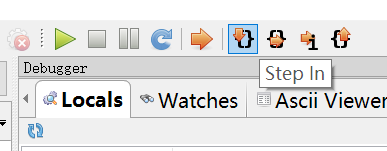


图18. Step in图标

如果想一下子执行到下一个断点，或者一下子把程序执行完，则选择绿色右向箭头“start/continue Debug”按钮即可。

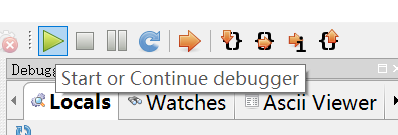


图19. 运行到下一个断点或者结束

**附录：**

可以使用如下代码建立项目，尝试单步跟踪进入(Step In)函数，对函数的实现进行调试。

|  |
| --- |
| #include <stdio.h>  int max(int x, int y);  int main()  {  int a,b;  int c;  scanf("%d,%d",&a,&b);    c= max(a,b); //设定断点，单步跟踪，并且跟踪到函数里面去  printf("c is %d\n", c);  return 0;  }  int max(int x, int y)  { int z; //调试的时候，可以查看x、y、z等变量的变化  if (x>y) z=x;  else z=y;  return z;  } |